EXHIBIT H

US005745908A

United States Patent [19]

Anderson et al.

[11] **Patent Number: 5,745,908**

[45] **Date of Patent: Apr. 28, 1998**

[54] **METHOD FOR CONVERTING A WORD PROCESSING FILE CONTAINING MARKUP LANGUAGE TAGS AND CONVENTIONAL COMPUTER CODE**

[75] Inventors: **Charles Anderson**, Newbury; **Stewart Barr**, Maidenhead; **Clive Beavis**, Wilts; **David Cleland**, Reading; **Raymond Obin**, Newbury, all of United Kingdom

[73] Assignee: **Systems Focus International**, Aruba, Netherlands

[21] Appl. No.: **625,743**

[22] Filed: **Mar. 29, 1996**

[51] **Int. Cl.**[6] **G06F 17/00**

[52] **U.S. Cl.** **707/513**

[58] **Field of Search** 395/774, 705, 395/752

[56] **References Cited**

U.S. PATENT DOCUMENTS

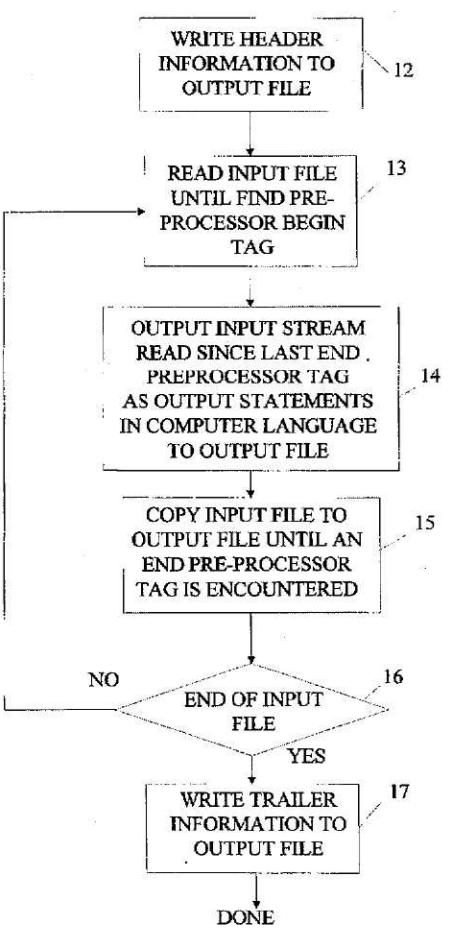
5,117,349	5/1992	Tirfing et al.	395/603
5,546,583	8/1996	Shriver	395/680
5,548,508	8/1996	Nagami	395/752
5,587,902	12/1996	Kugimiya	395/752

Primary Examiner—Joseph R. Burwell
Attorney, Agent, or Firm—Calvin B. Ward

[57] **ABSTRACT**

A method for operating a computer converts a word processing file to a computer program source file, which generates an output stream when run. The word processing file includes a linear sequence of characters having segments marked by beginning and ending tags. The method of the present invention translates the linear sequences of characters in the word processor file that are not surrounded by the tags to output statements in the computer program source file. The output statements cause the computer program generated from the source file, when executed, to reproduce the linear sequences of characters from the word processor file not surrounded by the tags in an output stream generated by the computer program. The method translates the linear sequences of characters in the word processing file that are surrounded by predetermined tags into computer source code statements. In so doing, the translation takes the form of merely copying the linear sequences of characters of the word processing file to the computer program source file or of translating the linear sequences of characters of the word processing file into statements recognized in the computer language.

6 Claims, 1 Drawing Sheet



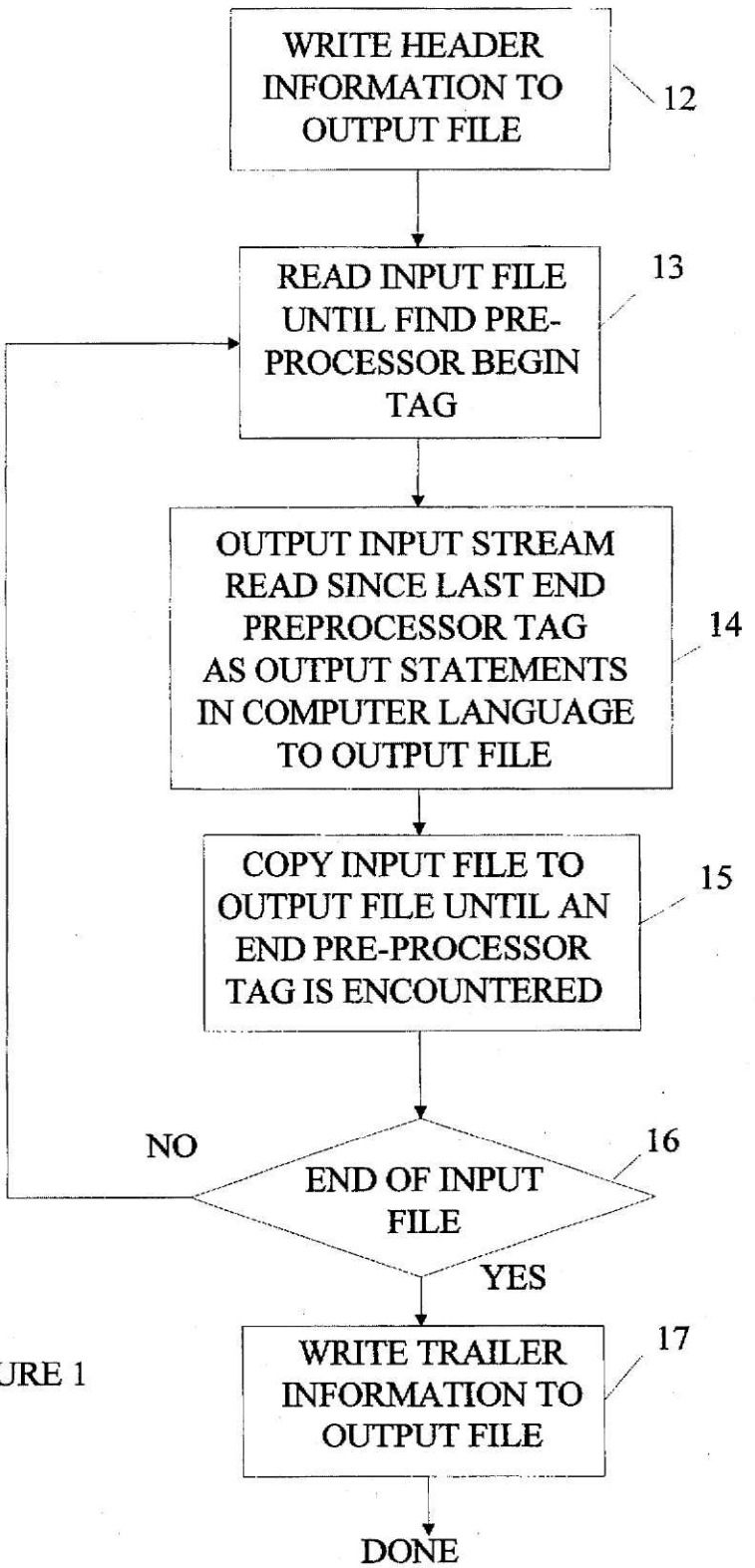

WRITE HEADER INFORMATION TO OUTPUT FILE
12
READ INPUT FILE UNTIL FIND PRE-PROCESSOR BEGIN TAG
13
OUTPUT INPUT STREAM READ SINCE LAST END PREPROCESSOR TAG AS OUTPUT STATEMENTS IN COMPUTER LANGUAGE TO OUTPUT FILE
14
COPY INPUT FILE TO OUTPUT FILE UNTIL AN END PRE-PROCESSOR TAG IS ENCOUNTERED
15
NO
END OF INPUT FILE
16
YES
WRITE TRAILER INFORMATION TO OUTPUT FILE
17
DONE

FIGURE 1

METHOD FOR CONVERTING A WORD PROCESSING FILE CONTAINING MARKUP LANGUAGE TAGS AND CONVENTIONAL COMPUTER CODE

FIELD OF THE INVENTION

The present invention relates to compilers for use in computing systems, and more particularly, to a preprocessor for generating a source file for a predetermined computer language compiler or interpreter in which the resulting computer program generates a word processing file that has part of its content generated by the computer program.

BACKGROUND OF THE INVENTION

The World Wide Web ("Web") has become a very successful means of communication between central sites connected to the Internet and individual users on the Internet who wish to communicate with the site. The communications are controlled by two programs, a Web Browser that runs on the user's computer and a Web server that runs on the site's computer. A Web Browser sends a request to a Web Server using the HTTP protocol. A request results in a MIME ("Multipurpose Internet Mail Extensions"—see IETF RFC1341, 1342, 1521) Stream being sent back to the Web Browser. The MIME stream includes a Content Type header for the data that indicates how the Web Browser will treat the data being sent. For example, a "text/html" MIME type indicates that the data is in the hypertext markup language (HTML), and should be interpreted accordingly; an "image/gif" MIME type indicates that the data is a "gif" image file, and should be rendered as an image after unpacking the data in the file.

The Web Server typically services a request either by sending back a file stored locally on the server or by running a program, the output of which is the MIME stream to be sent back to the browser. As noted above, the Web typically makes use of the hypertext format to display information to a user and receive input from the user. Hypertext allows a body of information to be organized into a hierarchical system in which the user can pursue increasing levels of specificity by following the various hypertext links from one document to the next. A typical hypertext text display system (a Web Browser) displays a document in which selected words or phrases are highlighted. The highlighted phrase indicates that another document related to that phrase is in the system. If the person viewing the document selects one of these words or phrases by pointing and clicking using a pointing device, the second document related to that word or phrase is sent to the user's screen. The user may return to the original document at any time selecting a "back" option on the viewer screen.

This form of information display has found wide acceptance on the Internet because of its ease of use. A user located at a terminal on the network connects to a server on the network that has a "home page" in hypertext format. The home page is then displayed on the user's screen by the browser. When the user selects a highlighted word, the browser communicates the user's choice to the server in a MIME data stream. The server then transfers the corresponding file to the user's machine via the network. The browser on the user's machine then displays this file to the user.

Conventional browser's also allow the user to input text on the user's screen which is then transferred to the server when the user selects a graphical element such as a "button".

Hence, the user can communicate information to the server beyond the predefined hypertext link information, provided the server is programmed to use this information.

The hypertext mode of information organization is also efficient from the point of view of the home page provider on the server. A home page is written in HTML. HTML is a word processing format which allows the user to define a page as the user would with a conventional word processor. In fact, programs for converting the various conventional word processing formats to HTML are commercially available. For each phrase that is to provide a link, the user marks the phrase by enclosing it with beginning and ending "tags". The user then defines another hypertext file that contains the document to be displayed in response to the user selecting the phrase. Hence, a server program can be as simple as a set of HTML documents created with a conventional word processing system and stored on the server.

If the interaction between the user and server is basically a transfer of predefined information which is static in nature, the simple "set of documents" mode is satisfactory. If, however, the information to be transferred requires some form of processing prior to the transfer, the simple hypertext engines are less than ideal. Consider an application in which the server must execute a program to gather and calculate the data that forms a portion of hypertext material that is to be returned to the user. To provide such a service, the server must include a program that is specific to the application and which performs the computations and then generates the results in the form of a hypertext document that is delivered on the network. The HTTP Protocol defines a general mechanism for programs to operate in this way, called the Common Gateway Interface (or CGI). A program that uses this mechanism is often referred to as a CGI Program.

There is no standardized program support for generating hypertext documents within a conventional processing program written in one of the common programming languages such as COBOL or C. The output facilities of these languages typically provide various forms of output statements that allow the program to output a block of text. Hence, to generate the hypertext document, the programmer must, in effect, include statements to writeout the hypertext portion of the document one line at a time. In addition, the user must be familiar with all of the various "tags" used in HTML so that the user can provide these tags in the output stream of the program.

Similarly, the hypertext generating programs provide, at most, a limited scripting capability to support computations on the server. For example, some engines have the ability to make calls to database engines to retrieve data to be inserted into the hypertext document.

Broadly, it is the object of the present invention to provide a system that combines the strength of both a word processor and a conventional computer language.

It is a further object of the present invention to provide a compiler that allows hypertext documents and native computer code to be combined in a manner in which the benefits of compiled general purpose computer language can be achieved in a hypertext document.

These and other objects of the present invention will become apparent to those skilled in the art from the following detailed description of the invention and the accompanying drawings.

SUMMARY OF THE INVENTION

The present invention is a method for operating a computer to convert a word processing file to a computer

program that will generate a word processing file when run. The word processing file includes a linear sequence of characters having segments marked by beginning and ending tags. The word processing file is converted to a source file for a computer language. The method of the present invention translates the linear sequences of characters in the word processor file that are not surrounded by the tags to output statements in the computer code source file. The output statements cause a computer program generated from the source file to reproduce the material in an output stream generated by the computer program. The method translates the linear sequences of the word processing file that are surrounded by predetermined tags into computer source code statements. The translation may take the form of merely copying the material to the source file or of translating the statements into statements recognized in the computer language.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flow chart for a preprocessor according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS OF THE INVENTION

The present invention may be more easily understood with reference to the structure of a typical prior art hypertext document. Such a document consists of a string of characters in which specific sequences of characters are tagged. For the purposes of this discussion, the tagged sequences can be viewed as consisting of two types, those related to on-screen formatting of the document and those related to defining links to other documents. The first type includes tags that mark the beginning and end of sequences to be displayed in a particular style which defines font, print size, etc. Since such formatting is only indirectly related to the present invention, it will not be discussed in detail here.

The second type defines the links to other files. These sequences are defined by placing predefined tags at the beginning and end of each sequence, and specifying the file to sent in response to a user selecting the sequence with the user's pointing device. As these tags are also well known in the computer arts, they will not be discussed further here.

A hypertext document according to the present invention may be viewed as a standard HTML file having a new class of tags. These new tags mark the beginning and end of code sequences in a conventional computer language such as COBOL. The modified document is then used as input to a pre-processor for the computer language compiler in question. The pre-processor converts all conventional hypertext material to output statements in the language in question. This conversion converts the hypertext to the underlying computer language. The converted file is then compiled in the conventional manner to generate an executable that is run on the server in response to a request for the hypertext file. This executable provides the MIME stream when the user requests the hypertext file.

The present invention utilizes pre-processing and compilation to avoid the need to parse the HTML file each time a request is made for the file. The alternative approach of reading the HTML file each time it is needed, interpreting the underlying computer language and replacing data variables places a significant computational load on the server.

The method of the present invention does not restrict either the HTML or computer language portion of the program. The computer language compiler processes the computer language statements, and the HTML browser processes the HTML statements.

The method of the present invention places no restrictions on the user's browser. The user's browser only needs to deal with standard HTML. There is no need to download a specific script language to the user's computer, and hence, the user's browser does not need to be customized for a specific scripting language.

To simplify the following discussion, the underlying computer language will be assumed to be COBOL; however, it will be apparent to those skilled in the art that the teachings of the present invention can be applied with a wide range of conventional computer languages.

In the preferred embodiment of the present invention, a COBOLTYPE tag is added to the HTML language tag set. This tag is optional, but if present, must be in the HTML stream, and must appear before any other HTML or COBOL tags. The COBOLTYPE tag identifies the MIME Content-Type of the output for this program. The full syntax for the COBOLTYPE tag is as follows:

<COBOLTYPE [CONTENT[-TYPE]=content-type]>

The CONTENT-TYPE parameter indicates the MIME type of the information to be output by the program in the correct format for a CGI Program. If this parameter is not specified, the content type will be "text/html". If the programmer wishes to prevent a content type being output (e.g., the program is not the first in a sequence of COBOL programs or the programmer has some special requirement) then the programmer can use the CONTENT-TYPE="" to ensure that no content type header is output.

The COBOL TAG may embedded anywhere in the HTML stream. The text following the <COBOL> tag is assumed to be COBOL source code until a </COBOL> tag is encountered. For convenience a comment tag <!--COBOL statements--> is also included to allow COBOL statements to be inserted.

In the preferred embodiment of the present invention, an <ITERATE> tag is also included. The <ITERATE> tag marks the beginning of an iteration over a table of variable values. The full syntax for the tag is <ITERATE [OVER] variableName>. The number of iterations is the number of the "Occurs" parameter specified for variableName. The iteration continues until a </ITERATE>. Tag is found.

In the preferred embodiment of the present invention, there is also a variable replacement tag having the form <%name%> or alternatively %%name%%. The variable called "name" is substituted for this tag wherever the tag is found. The replacement occurs everywhere in the document, outside the COBOL tagged portion. The replacement does not take place within COBOL procedural statements. If <%name%> occurs within an iteration block, the number of the current iteration will be used to determine which occurrence of the variable is to be used. If the program is running as a CGI Program, then the variable name refers to the name of the variable received in the HTTP GET or POST stream. If the program has been called by another program passing a parameter containing replacement variables, then the variable name refers to the variable received in the linkage section parameter as discussed below. While the variable replacement operation could be implemented with separate COBOL, sections without the need to define a new tag, the tags are preferred since they require much less coding by the programmer.

In the preferred embodiment of the present invention, there is also a language variable replacement tag having the

form <$dataname$> or alternatively $$dataname$$. The language variable called "dataname" is substituted for this tag wherever the tag is found. The replacement occurs everywhere in the document, outside the COBOL tagged portion. The replacement does not take place within COBOL procedural statements. The variable called dataname must be declared as a language variable in the hypertext COBOL program. While the language variable replacement operation could be implemented with separate COBOL sections without the need to define a new tag, the tags are preferred since they require much less coding by the programmer.

As noted above, if the program is called by another program which passes the variables that are to be applied to the embedded COBOL program, these variables are accessed within the COBOL program in the same way as variables received from the HTTP GET or POST stream. The calling program (if any) may pass the following parameter:

```
01   HTMLMAP-Parameter
     03   VariableNameLen      pic 9(2) comp-5.     *> length of variable name
     03   VariableName.                             *> variable name
```

8

```
    05   PIC X OCCURS 1 TO 255 DEPENDING ON VariableNameLen.
03  ValueLen            pic 9(4) comp-5.     *> length of data item
03  MaxOccurs           pic 9(9) comp-5.     *> Maximum possible OCCURS (=1 if
                                                variable is not a table)
03  ActualOccurs        pic 9(9) comp-5.     *> Actual OCCURS (=1 if not a
                                                table)
03  Stride              pic 9(9) comp-5.     *> Number of bytes in each table
                                                item
03  ValuePtr            pointer.             *> Pointer to actual data area
```

The above discussion can be more easily understood with reference to a simple COBOL-hypertext program. The program adds information to a database and displays the added information to the user. The hybrid file created by the programmer is as follows:

15

```
<COBOL>
select F1 assign "register.dat"
    organization is indexed
    access mode is dynamic
    record key is fEmail
    alternate key is fOrganization.

file section.
fd  f1.
01  f1-Rec.
    03  fName          pic x(60).
    03  fEmail         pic x(60).
    03  fOrganization  pic x(60).
    03  fAddress       pic x(250).
    03  fPhone         pic x(20).
    03  fFax           pic x(20).
    03  fWhen          pic x(8).
    03  fIP            pic x(15).
    03  fHost          pic x(30).
```

9

```
working-storage section.
01  DateTime.
    03  UnformattedDate.
        05  UYear       pic 9(4).
        05  UMonth      pic 9(2).
        05  UDay        pic 9(2).

    03  FormattedDate.
        05  ODayOfMonth  pic z9.
        05               pic x.
        05  OMonthWord   pic x(3).
        05               pic x.
        05  OFullYear    pic 9(4).

    03  MonthNames      pic x(36) value
        "JanFebMarAprMayJunJulAugSepOctNovDec".
    03  redefines MonthNames.
        05  MonthName   pic x(3) occurs 12.

</COBOL>
<HTML>
<HEAD>
<TITLE>Registration Database: Information Gathered</TITLE>

<!
-- set up date info, this is an SGML comment
--
--COBOL
move function current-date to UnformattedDate
move UYear to OFullYear
--Now deal with the month
  (you can have 2 hyphens at neither, either or both ends of this line,
                                                    parser is forgiving)
--COBOL move MonthName(UMonth) to OMonthWord
       move UDay to ODayOfMonth --
```

10

```
-- all done --
>

<BODY>
<H1>Registration <!--HTML Comment, just for fun-->Confirmation</h1>
Received on $$FormattedDate$$ from %%HTTP_USER_AGENT%%
acting for %%REMOTE_HOST%% (%%REMOTE_ADDR%%)
<hr>

<!--COBOL

getValue "Name" fName
getValue "Email" fEmail
getValue "Organization" fOrganization
getValue "Address" fAddress
getValue "Phone" fPhone
getValue "Fax" fFax
-->
<COBOL> getValue "REMOTE_HOST" fIP </COBOL>
<COBOL> getValue "REMOTE_ADDR" fHost
move UnformattedDate to fWhen

open i-o F1

write F1-Rec </COBOL>

The following information has been filed:

<table>
<tr>
<td>Name<td><b>%%Name%%</b>
<td>Email<td><b>%%Email%%</b>
<tr>
<td>Organization<td><b>%%Organization%%</b>
<tr>
```

```
<td>Phone<td><b>%%Phone%%</b>
<td>Fax<td><b>%%Fax%%</b>
<tr>
<td>Address<td><b>%%Address%%</b>

</table>
```

This hybrid source file is converted by the COBOL pre-processor to a file that may be viewed as having a header/trailer which provides the interface to the MIME stream on the server and a COBOL section in which the material within the COBOL tags is reproduced and the material within the hypertext sections is converted to COBOL display statements. The header section is as follows:

```
$set sourceformat(free) mfoo odoslide
*>-------------------------------------------------------
*> LiveCOBOL support file. *** DO NO AMEND ***
*>
*> These declarations go at the start of the program
*>
*>
program-id. "ShowVariable" is external.
linkage section.
01  lcFieldName   pic x(255).
procedure division using lcFieldName delimited
invoked as == ShowValue <lcFieldName> ==.
end program "ShowVariable".

program-id. "ShowOccurence" is external.
linkage section.
01  lcFieldName   pic x(255).
01  lcIterator    pic 9(9) comp-5.
procedure division using lcFieldName delimited lcIterator
invoked as == ShowOccurence <lcFieldName> <lcIterator> ==.
end program "ShowOccurence".
```

```
program-id. "Show" is external.
linkage section.
01  lcFieldName   pic x(255).
01  ReturnChar    pic x.
procedure division using lcFieldName delimited
                 returning ReturnChar
invoked as function == Show ( <lcFieldName> ) ==.
end program "Show".

program-id. "GetLength" is external.
linkage section.
01  lcFieldName   pic x(255).
01  lcLength      pic 9(9) comp-5.
procedure division using lcFieldName delimited lcLength
invoked as == GetLength <lcFieldName> <lcLength> ==.
end program "GetLength".

program-id. "GetValue" is external.
linkage section.
01  lcFieldName   pic x(255).
01  lcBuffer      pic x(500).
procedure division using lcFieldName delimited lcBuffer
invoked as == GetValue <lcFieldName> <lcBuffer> ==.
end program "GetValue".

program-id. "GetOccurence" is external.
linkage section.
01  lcFieldName   pic x(255).
01  lcIterator    pic 9(9) comp-5.
01  lcBuffer      pic x(500).
procedure division using lcFieldName delimited lcIterator lcBuffer
invoked as == GetOccurence <lcFieldName> <lcIterator> <lcBuffer> ==.
end program "GetOccurence".

program-id. "GetOccursCount" is external.
```

```
linkage section.
01  lcFieldName    pic x(255).
01  lcIterator     pic 9(9) comp-5.
procedure division using lcFieldName delimited lcIterator
invoked as == GetOccursCount <lcFieldName> <lcIterator> ==.
end program "GetOccursCount".

program-id. "SayValue" is external.
linkage section.
01  lcBuffer          pic x(65535).
procedure division using lcBuffer delimited
invoked as == DISPLAYZ <lcBuffer> == .
end program "SayValue".

program-id. "ShowValue" is external.
linkage section.
01  lcBuffer          pic x(65535).
01  ReturnChar        pic x.
procedure division using lcBuffer delimited
                 returning ReturnChar
invoked as function == Z ( <lcBuffer> ) == .
end program "ShowValue".
*>-------------------------------------------------------
```

The actual input file is then converted to the following file:

```
            id division.
            program-id. registernow.
        select F1 assign "register.dat"
            organization is indexed
            access mode is dynamic
            record key is fEmail
            alternate key is fOrganization.
```

14

```
file section.
fd  f1.
01  f1-Rec.
    03  fName          pic x(60).
    03  fEmail         pic x(60).
    03  fOrganization  pic x(60).
    03  fAddress       pic x(250).
    03  fPhone         pic x(20).
    03  fFax           pic x(20).
    03  fWhen          pic x(8).
    03  fIP            pic x(15).
    03  fHost          pic x(30).

working-storage section.
      *>-----------------------------------------------------------------
      *> LiveCOBOL support file. *** DO NO AMEND ***
      *>
      *> These declarations go in WORKING-STORAGE SECTION
      *>
      *>
      78  CGI-INPUT-MODULE           value "acccgi.int".

      01  xA7-func                   pic x comp-x.

      01  xA7-parm                   pic x comp-x.

      01  LiveCOBOLVariables.

          *>  variableLen can be used to hold lengths
          03  variableLen            pic 9(9) comp-5.

          *>  iterator holds the current iteration number.
          03  iterator               pic 9(9) comp-5.

          *>  occursCount holds the number of iterations
          03  occursCount            pic 9(9) comp-5.
```

15

```
           *>  buffer can be used to hold alphanumeric data
           03  buffer                  pic x(500).

       01  WorkingVariables.
           *>  HTMLMap-Base points to the base of the HTMLMAP structure
           03  HTMLMap-Base            pointer.

           *>  Temporary length
           03  lcTempLength            pic 9(9) comp-5.
           *>  Temporary pointer
           03  lcTempPointer           pointer.

           03  NameLength              pic 9(9) comp-5.
           03  CGIIndex                pic 9(9) comp-5.
           03  CGIIndex2               pic 9(9) comp-5.
           03  ValueStart              pic 9(9) comp-5.
           03  ValueLength             pic 9(9) comp-5.
           03  ValueCount              pic 9(9) comp-5.
           03  LiveMode                pic 9(2) comp-5.
               88  Standalone          value 0.
               88  HTMLMapMode         value 1.
           03  VariableMode            pic 9(2) comp-5.
               88  VarHTMLMap          value 0.
               88  VarEnvironment      value 1.

       *>  Space for the HTMLMap data.
       *>  Note that the size of this must be large enough for all the
       *>  variables that will be contained in the HTMLMap.
       *>  See ONIONS4.IF for details of the HTMLMap-Parameter.
       01  HTMLMap                 pic x(1024).
       01  CGIChar redefines HTMLMap pic 9(2) comp-x occurs 1024.

       *>-----------------------------------------------------------
```

16

```
01  DateTime.
    03  UnformattedDate.
        05  UYear        pic 9(4).
        05  UMonth       pic 9(2).
        05  UDay         pic 9(2).

    03  FormattedDate.
        05  ODayOfMonth  pic z9.
        05                pic x.
        05  OMonthWord    pic x(3).
        05                pic x.
        05  OFullYear     pic 9(4).

    03  MonthNames       pic x(36) value
        "JanFebMarAprMayJunJulAugSepOctNovDec".
    03  redefines MonthNames.
        05  MonthName    pic x(3) occurs 12.


        local-storage section.
        linkage section.
       *>-----------------------------------------------------------------
       *> LiveCOBOL support file. *** DO NO AMEND ***
       *>
       *> These declarations will be placed in LINKAGE SECTION.
       *>
       *> Parameters used by the Vocabulary routines

        01  lcFieldName          pic x(255).
        01  lcLength             pic 9(9) comp-5.
        01  lcBuffer             pic x(65535).
        01  lcTempBuffer         pic x(65535).
        01  lcIterator           pic 9(9) comp-5.

       *> The HTMLMap structure
        01  HTMLMAP-Parameter.
```

17

```
*>  length of the variable name
03  variableNameLen    pic 9(2) comp-x.
*> n is the value of variableNameLen
03  variableName.
    05        pic x(1) occurs 0 to 1
                       depending on variableNameLen.
*> length of the data item
03  valueLen           pic 9(9) comp-x.
*> maximum possible OCCURS (=1 if not a table)
03  maxOccurs          pic 9(9) comp-x.
*> actual OCCURS (=1 if not a table)
03  actualOccurs       pic 9(9) comp-x.
*> number of bytes from start of one item to the next
03  stride             pic 9(9) comp-x.
*> pointer to the actual data area
03  valuePtr           pointer.

*>-----------------------------------------------------------------
*>-----------------------------------------------------------------
*> LiveCOBOL support file. *** DO NO AMEND ***
*>
*> These declarations will be placed at the start procedure div.
*>
*>
    PROCEDURE DIVISION USING HTMLMap-Parameter.
    set HTMLMap-Base to address of HTMLMap-Parameter
    if HTMLMap-Base = null
        set HTMLMap-Base to address of HTMLMap
        set Standalone to true
    else
        set HTMLMapMode to true
    end-if

    if Standalone
        call CGI-INPUT-MODULE
        call "CBL_ACCEPT_CGI" using HTMLMap
```

18

```
                move 18 to xA7-func
                move  1 to xA7-parm
                call x"A7" using xA7-func
                                 xA7-parm
                end-call
            end-if
        *>-------------------------------------------------------------
             display "Content-type :" "text/html"
                 & X"0d0a"
display
    "<HTML>"
display
    "<HEAD>"
display
    "<TITLE>Registration Database: Information Gathered</TITLE>"
            display ' '


move function current-date to UnformattedDate
move UYear to OFullYear


        move MonthName(UMonth) to OMonthWord
        move UDay to ODayOfMonth


            display ' '
display
    "<BODY>"
display
    "<H1>Registration "


display
    "Confirmation</h1>"
display
```

19

```
    "Received on "
FormattedDate
    " from "
     with no advancing
     ShowValue "HTTP_USER_AGENT"
display
    " "
display
    "acting for "
     with no advancing
     ShowValue "REMOTE_HOST"
display
    " ("
     with no advancing
     ShowValue "REMOTE_ADDR"
display
    ")"
display
    "<hr>"
            display ' '


getValue "Name" fName
getValue "Email" fEmail
getValue "Organization" fOrganization
getValue "Address" fAddress
getValue "Phone" fPhone
getValue "Fax" fFax

        getValue "REMOTE_HOST" fIP

        getValue "REMOTE_ADDR" fHost
move UnformattedDate to fWhen

open i-o F1
```

20

```
write F1-Rec

            display ' '
            display ' '
            display ' '
display
    "The following information has been filed:"
            display ' '
display
    "<table>"
display
    "<tr>"
display
    "<td>Name<td><b>"
     with no advancing
     ShowValue "Name"
display
    "</b>"
display
    "<td>Email<td><b>"
     with no advancing
     ShowValue "Email"
display
    "</b>"
display
    "<tr>"
display
    "<td>Organization<td><b>"
     with no advancing
     ShowValue "Organization"
display
    "</b>"
display
    "<tr>"
display
    "<td>Phone<td><b>"
```

```
        with no advancing
        ShowValue "Phone"
    display
        "</b>"
    display
        "<td>Fax<td><b>"
        with no advancing
        ShowValue "Fax"
    display
        "</b>"
    display
        "<tr>"
    display
        "<td>Address<td><b>"
        with no advancing
        ShowValue "Address"
    display
        "</b>"
                display ' '
    display
        "</table>"
           goback.
```

The trailer is as follows:

```
*>-------------------------------------------------------------
*> LiveCOBOL support file. *** DO NO AMEND ***
*>
*> These declarations will be placed at the end of the program
*>
*>
entry "GetLength" using lcFieldName lcLength.

    move 0 to lcLength
    set VarEnvironment to true
```

22

```
        perform SetupName

        if Standalone
            perform FindCGI-Data
        else
            perform FindHTMLMap-Data
        end-if

        if varEnvironment
            display lcFieldName(1:NameLength) upon environment-name
            accept buffer from environment-value
            move length of buffer to ValueLength
            perform until ValueLength = 0 or buffer(ValueLength:1) not = space
                subtract 1 from ValueLength
            end-perform
        end-if

        move ValueLength to lcLength

        exit program.

    entry "GetValue" using lcFieldName lcBuffer.

        set VarEnvironment to true
        perform SetupName
        move x"00" to lcBuffer(1:1)

        if Standalone
            perform FindCGI-Data
            if VarHTMLMap
                move HTMLMap(ValueStart:ValueLength) to
                                lcBuffer(1:ValueLength)
            end-if
        else
            perform FindHTMLMap-Data
            if VarHTMLMap
```

```
                move lcTempBuffer(1:ValueLength) to
                                lcBuffer(1:valueLength)
            end-if
        end-if

        if varEnvironment
            display lcFieldName(1:NameLength) upon environment-name
            accept Buffer from environment-value
            move length of buffer to ValueLength
            perform until ValueLength = 0 or buffer(ValueLength:1) not=space
                subtract 1 from ValueLength
            end-perform
            if ValueLength <> 0
                move buffer(1:ValueLength) to lcBuffer(1:ValueLength)
            end-if
        end-if

        if ValueLength <> 0
            move x"00" to lcBuffer(ValueLength + 1:1)
        end-if

        exit program.

    entry "GetOccurence" using lcFieldName lcIterator lcBuffer.

        set VarEnvironment to true
        perform SetupName
        move x"00" to lcBuffer(1:1)
        move lcIterator to ValueCount

        if Standalone
            perform FindCGI-Data
            if VarHTMLMap
                move 0 to ValueCount
                perform until exit
                    if (CGIIndex > length of HTMLMap) or
```

```
                (HTMLMap(CGIIndex:1) = X"00")
                exit program
            end-if
            if  (CGIChar(CGIIndex) = NameLength) and
                                            *> Name Length Match
                (HTMLMap(CGIIndex + 1:NameLength) =
                                        lcFieldName(1:NameLength))
                                                *> Name match
                add CGIChar(CGIIndex) to CGIIndex *> Skip over name
                add 1                 to CGIIndex
                add 1                 to ValueCount *> Value Count
                if ValueCount = lcIterator
                    move CGIChar(CGIIndex) to ValueLength
                                                *> Value Size
                    add 1             to CGIIndex
                    move HTMLMap(CGIIndex:ValueLength)
                        to lcBuffer(1:ValueLength)
                    move x"00"        to lcBuffer(ValueLength + 1:1)
                    exit program
                end-if
            else
                add CGIChar(CGIIndex) to CGIIndex
                add 1 to CGIIndex
            end-if
            add CGIChar(CGIIndex) to CGIIndex *> Skip over value
            add 1                 to CGIIndex
        end-perform
    end-if
else
    perform FindHTMLMap-Data
    if VarHTMLMap
        if lcIterator <= actualOccurs
            set lcTempPointer to valuePtr
            compute lcTempLength = stride * (lcIterator - 1)
            set lcTempPointer up by lcTempLength
            set address of lcTempBuffer to lcTempPointer
```

25

```
                perform TrimSpaces
                move lcTempBuffer(1:ValueLength) to
                                            lcBuffer(1:ValueLength)
                move x"00" to lcBuffer(ValueLength + 1:1)
            end-if
        end-if
    end-if

    exit program.

entry "GetOccursCount" using lcFieldName lcIterator.

    move 0 to lcIterator
    set VarEnvironment to true
    perform SetupName

    if Standalone
        perform FindCGI-Data
        if VarHTMLMap
            move 0 to lcIterator
            perform until exit
                if (CGIIndex > length of HTMLMap) or
                        (HTMLMap(CGIIndex:1) = X"00")
                    exit program
                end-if
                if  (CGIChar(CGIIndex) = NameLength) and
                                            *> Name Length Match
                    (HTMLMap(CGIIndex + 1:NameLength) =
                                                lcFieldName(1:NameLength))
                                                        *> Name match
                    add 1                to lcIterator   *> Value Count
                end-if
                add CGIChar(CGIIndex) to CGIIndex *> Skip over name
                add 1                 to CGIIndex
                add CGIChar(CGIIndex) to CGIIndex *> Skip over value
                add 1                 to CGIIndex
```

```
            end-perform
        end-if
    else
        perform FindHTMLMap-Data
        if VarHTMLMap
            move actualOccurs to lcIterator
        end-if
    end-if

    exit program.

entry "ShowOccurence" using lcFieldName lcIterator.
    GetOccurence lcFieldName lcIterator buffer
    displayz buffer
    exit program
    .

entry "SayValue" using lcBuffer.
entry "ShowValue" using lcBuffer.
    perform varying ValueLength from 1 by 1 until
        (ValueLength > 128) or
        (lcBuffer(ValueLength:1) = X"00")
    end-perform
    subtract 1 from ValueLength
    display lcBuffer(1:ValueLength) with no advancing
    exit program returning 8224.

entry "ShowVariable" using lcFieldName.
entry "Show" using lcFieldName.
    *>GetValue lcFieldName buffer

    set VarEnvironment to true
    perform SetupName
    move x"00" to buffer(1:1)

    if Standalone
```

```
            perform FindCGI-Data
            if VarHTMLMap
                move HTMLMap(ValueStart:ValueLength) to buffer
            end-if
        else
            perform FindHTMLMap-Data
            if VarHTMLMap
                move lcTempBuffer(1:ValueLength) to buffer(1:ValueLength)
            end-if
        end-if

        if varEnvironment
            display lcFieldName(1:NameLength) upon environment-name
            accept buffer from environment-value
            move length of buffer to ValueLength
            perform until ValueLength = 0 or buffer(ValueLength:1) not=space
                subtract 1 from ValueLength
            end-perform
        end-if

        if ValueLength <> 0
            move x"00" to buffer(ValueLength + 1:1)
            display buffer(1:ValueLength) with no advancing
        end-if

        exit program returning 8224.

    *> given lcFieldName, return the length of the fieldname in NameLength

    SetupName.
        perform varying NameLength from 1 by 1 until
            (NameLength > length of lcFieldName) or
            (lcFieldName(NameLength:1) = X"00")
        end-perform
        subtract 1 from NameLength
        call "CBL_TOUPPER" using lcFieldName
```

```
                    by value NameLength
    end-call.

*> given lcFieldName and NameLength (length of fieldname)
*> locates the correct slot in HTMLMAP-Parameter

FindCGI-Data.

   move 1 to CGIIndex
   perform until exit
       if (CGIIndex > length of HTMLMap) OR
              (HTMLMap(CGIIndex:1) = X"00")
           move 0 to ValueLength
           exit perform
       end-if
       if  (CGIChar(CGIIndex) = NameLength) AND  *> Name Length Match
           (HTMLMap(CGIIndex + 1:NameLength) = lcFieldName(1:NameLength))
                                              *> Name match
           move CGIIndex to CGIIndex2
           add CGIChar(CGIIndex2)  to CGIIndex2 *> Skip over name
           add 1                   to CGIIndex2
           move CGIChar(CGIIndex2) to ValueLength   *> Value Size
           add 1                   to CGIIndex2
           move CGIIndex2 to ValueStart
       *>  add ValueLength to CGIIndex
           set VarHTMLMap to true
           exit perform
       else
           add CGIChar(CGIIndex)  to CGIIndex *> Skip over name
           add 1                  to CGIIndex
           add CGIChar(CGIIndex)  to CGIIndex *> Skip over value
           add 1                  to CGIIndex
       end-if
   end-perform
```

```
*> given lcFieldName and NameLength (length of fieldname)
*> locates the correct slot in HTMLMAP-Parameter

FindHTMLMap-Data.
    set address of HTMLMap-Parameter to HTMLMap-Base
    set lcTempPointer to HTMLMap-Base

    move 0 to ValueLength
    perform until variableNameLen = 0
        if variableNameLen = NameLength     *> Length match
            if variableName(1:NameLength) = lcFieldName(1:NameLength)
                set address of lcTempBuffer to valuePtr
                perform TrimSpaces
                set VarHTMLMap to true
                exit perform
            end-if
        end-if
        compute lcTempLength = 20 + variableNameLen + 1
        set lcTempPointer up by lcTempLength
        set address of HTMLMap-Parameter to lcTempPointer
    end-perform
    .

TrimSpaces.
    move valueLen to ValueLength
    perform until ValueLength = 0
        if lcTempBuffer(ValueLength:1) not = space
            exit perform
        else
            subtract 1 from ValueLength
        end-if
    end-perform
    .
*>-------------------------------------------------------------
```

```
end program registernow.
```

While the above description of the present invention has been in terms of HTML and COBOL, it will be apparent to those skilled in the art that that the teachings of the present invention may be applied to any word processing format and computer language. The word processing file format is modified by defining a beginning and ending tag for the computer language segment.

The pre-processor for the computer language creates a source code file for the compiler in which output statements (e.g. in a COBOL implementation, DISPLAY statements are used) that, when executed, reproduce everything that is not enclosed by the language tags are inserted for the word processing file material. The code between the tags is copied to the source file. The executable file generated by the source code when compiled with the appropriate compiler or interpreter will then generate a word processing file in the word-processing format when executed.

Refer now to FIG. **1** which is a flow chart for a preprocessor according to the present invention. The pre-processor starts by writing any header information needed by the compiler or the operating system interface to the output file as shown at **12**. This information may include the code needed to access particular input variables such as described above with respect to the HTML example.

The pre-processor then reads the input file until it finds a pre-processor tag that begins a computer language segment as shown at **13**. The material from the input file that has been read prior to encountering the starting language tag is written to the output file as display statements in the computer language as shown at **14**. The actual output sequence is a matter of design choice. For example, each line can be written as a single display statement as the line is read from the input file if the line does not contain a tag. Alternatively, the entire input stream from the input file since the last end tag can be stored and processed as a bloc in a single display statement. For the purposes of this discussion, a display statement is defined as any statement which will reproduce the material when executed in the resulting computer program.

The material from the beginning pre-processor tag to the ending pre-processor tag is then transferred to the output file as shown at **15**. Any desired code translation can also be accomplished here. For example, the display tags described above with reference to the HTML embodiments of the present invention are expanded in terms of source language statements as part of this process.

When the end tag corresponding to the beginning pre-processor tag is encountered, the program checks for additional input. If the end of the input file is encountered, any trailers needed by the compiler and interface code are added to the output file as shown at **17**. If there is more material in the input file, the program resumes reading the input file at block **13**.

It should be noted that the pre-processor does not have to "understand" the word processor file formats. The pre-processor only needs to recognize the computer language tags. Any tag format can be used provided the word processing file format, or the embedded document, does not contain a tag of this form. Similarly, the word processing software does not have to "understand" the computer language tags; hence, no modifications to the word processing software are required. Finally, the computer language compiler also does not need to be modified.

In general, additional tags can be added to the definition. The replacement tags defined above are examples of such tags. Again, only the preprocessor needs to understand these tags. In addition, some form of interface tag that defines the inputs and output, i.e., a file, console, particular data stream, etc., will in general be useful.

It should also be noted that the present invention provides a means for defining a "live" document that may be easily transported between platforms. The source code provided by the pre-processor may be used on any platform that provides a compiler for that source code.

While the above described embodiments have referred to compiled computer languages, it will be apparent to those skilled in the art that the teachings of the present invention can be applied to interpreted languages such as BASIC and REXX.

The above described embodiments of the present invention have referred to a system in which the compiled computer language file resides on the server and is executed when a request for the "document" is received. However, it will be apparent to those skilled in the art from the above discussion that the source file may be stored instead. In such a system, the source file would be compiled the first time a request for the document is received. The operating system on the server can detect the need to re-compile the file by checking the time stamps on the source file and the executable file currently stored on the server. If the source file has been altered since the last compilation, the operating system causes the source file to be re-compiled prior to invoking the executable file.

Various modifications to the present invention will become apparent to those skilled in the art from the foregoing description and accompanying drawings. Accordingly, the present invention is to be limited solely by the scope of the following claims.

What is claimed is:

1. A method for operating a computer to convert a word processing file, which comprises a linear sequence of characters, said linear sequence of characters comprising segments of said linear sequence marked by beginning and ending tags, into a source file comprising computer language statements, said source file defining a computer program that will generate an output stream in a format recognizable by a word processing program adapted to read said output stream, said method comprising the steps of:

reading a linear sequence of characters from said word processing file;

translating a segment of said linear sequence of characters that is not surrounded by said beginning and ending tags to computer language output statements and outputting said computer language output statements to said source file,

wherein said output statements cause a computer program generated from said source file, when said generated computer program is executed, to reproduce said linear sequence of characters in an output stream generated by said computer program; and

providing computer source language statements in said source file corresponding to a segment of said linear sequence of characters that is surrounded by said beginning and ending tags.

2. The method of claim **1** wherein said word processing file comprises HTML.

3. The method of claim **1** wherein said computer language is COBOL.

4. The method of claim **1** wherein said tags occur in pairs, one marking the beginning of a sequence of characters and one marking the end of a sequence of characters, and wherein said pairs of tags are of first and second types, said

first pair of tags marking sequences of characters that are to be copied to said source file without alteration, and said pair of tags marking sequences of characters that are to be translated to predetermined source code statements determined by an indication associated with said second pair of tags.

5. The method of claim **1** further comprising the step of providing computer source code statements in said source file enabling a program generated from said source file to receive input from a source outside of said generated computer program.

6. The method of claim **1** further comprising the step of replacing predetermined syntax in said linear sequence of characters that is not surrounded by said tags with variables defined outside of said linear sequence of characters.

* * * * *

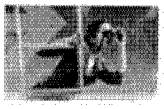
EXHIBIT I

SUBMITTED UNDER SEAL PURSUANT TO PROTECTIVE ORDER

EXHIBIT J

10 tips for finding information on the Internet

By Marc Saltzman



Sometimes looking for quick information on the Web can be like searching for a needle in a haystack. With so many billions of Web pages in cyberspace, finding specific information can be a daunting task.

"Often when I use search engines I get so many irrelevant results that I just give up," admits a frustrated Lorraine Adams, mother of two and a disability consultant.

Perhaps Adams isn't aware that there are ways to get more out of your favorite search engine, whether that's Live Search, Google, or Yahoo!. A few searching tips, tricks, tweaks, and techniques can help you find what you are looking for in cyberspace quickly and easily.

Follow these 10 suggestions with your favorite search engine.

1. Use the advanced search field

Almost all search engines have an "advanced search" area that provides Web surfers with more specific options. Here, you can search by an update date, look for Web sites with a specific domain like ".net," or find Web sites in a preferred language.

Top of page

2. Search with a phrase

To better help the search engine find what you're looking for, offer a sequence of words in a specific order, using quotation marks around the phrase.

For instance, if you're looking for information on the TV show *Saturday Night Live*, type "*Saturday Night Live*" into the search window. Without quotations on each end, a search engine will likely look for Web sites containing any of the words separately: *Saturday*, *night*, and *live*.

Top of page

3. Be specific

If broad search words like car classifieds yield too many results, try more specific words such as *used car classifieds*, *Mercedes classifieds*, or *London car classifieds*.

Top of page

4. Use alternative search words

If your keywords do not produce the results you want, try synonyms. Use a thesaurus like Roget's Interactive Thesaurus or Merriam-Webster OnLine to find alternative search words. After all, a dog is also a canine, a pooch, a mutt, a hound, a pet, and man's best friend!

Top of page

5. Insert a plus or minus sign

This trick usually works in most search engines. Put a plus sign (+) in front of a word that must be found in the search window. For example, *city guides + New York* will help you narrow the search for city guides for New York only.

Similarly, place a minus sign (-) in front of a word that you do not want to appear in the search results. Typing in *python -Monty* will allow you to come up with results for snakes and not the British comedy troupe.

Top of page

6. Just search the domain name

If you know the Web site that you want to search, but aren't sure where the information is located within that site, you can tell your search engine to only search that domain. Enter what you're looking for in the search field, followed by the word "site" and a colon, and then by the domain name.

For example, to find admission information for the University of Toronto, enter this: *admission site:www.utoronto.ca*.

Top of page

7. Explore "best of" sites

If you're after quality and not quantity, a few search engines provide access to hand-picked "Best of the Web" sites. One example is About.com.

Top of page

8. Eliminate inappropriate content

If you prefer not to have adult sites included in your search results, activate SafeSearch on the Live Search settings page or on Google's advanced search page. While not 100 percent accurate, this function will eliminate sites that contain explicit sexual content from the search results.

Top of page

9. Save time with a search toolbar

If you do a lot of searching on the Web, consider downloading the free Windows Live Toolbar or Google Toolbar, which always sit near the top of your Internet Explorer browser window. This way, you do not need to leave the Web site you're on to type in a new query.

Top of page

10. Specialize your search engine

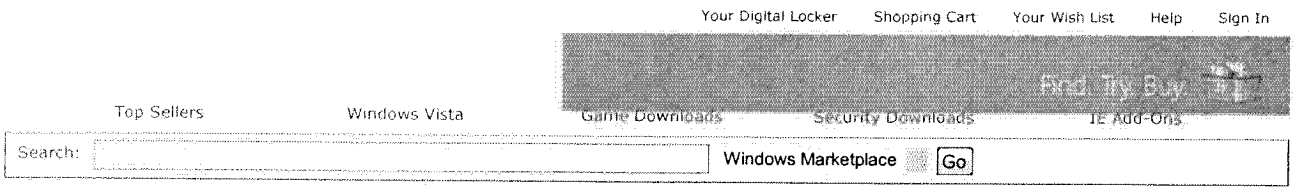
Lastly, keep in mind that there are many specialized search engines. Live Search, for example, has Live Search Images to search for pictures, MSN Shopping for shopping-related Web sites, and MSN News for published news articles from around the world. Many more specialty search engines can be found at Search Engine Watch.

Article written by Marc Saltzman and adapted from an original piece from Microsoft Home Magazine.

Top of page

Related Links

- 5 ways to keep in touch online
- Keep your Web browser running like new
- How and why to clear your cache


Your Digital Locker
Shopping Cart
Your Wish List
Help
Sign In
Find Try Buy
Top Sellers
Windows Vista
Game Downloads
Security Downloads
IE Add-Ons
Search:
Windows Marketplace
Go

Google Toolbar For IE By Google

Customer Rated Rate this item >

More Product views:

Google Toolbar enhances Internet Explorer with a Google search box and other useful tools like a pop-up blocker, web form filler, and spellchecker. With the new Google Toolbar beta, you can add buttons and bookmarks, get instant search suggestions, and share web pages with friends via email, blog, or SMS. Google Toolbar is free and installs in seconds. Version 4.0.1020.3054 may include unspecified updates, enhancements, or bug fixes.

View full product specification

Purchasing Options

Digital Locker Download
FREE from Tucows

What is this?

Demos and Trials

Download Free Version
From CNET Download.com

Add to Wish List

Detailed Product Specifications

Brand	Google
Minimum Requirements - Supported OS	Windows 2000 Windows XP
Minimum Requirements-Dependencies	Internet Explorer 6
Miscellaneous-License Type	Free
Miscellaneous-Limitations	No limitations
Miscellaneous-Uninstaller Included	Yes
Sizespeed - Download Size	817.56KB
Sizespeed - Download Speed	Dial-Up (53.3 kbps): 00:02:03|DSL/Cable (384 kbps): 00:00:17|DSL/Cable (768 kbps): 00:00:09|T1 (1.5 Mbps): 00:00:04

Customer Reviews

Rate this item

Drag your mouse across the stars to select a rating from 1 to 5 stars, and then click to save it.
To avoid duplicate ratings, we will ask you to sign in using a Windows Live ID.

Review this Product

WRITE A REVIEW

Showing 1 of 1 Customer Reviews

freddie400

By freddie400
11/8/2007

Pros:
Is Good

Cons:

Comments:

47 of 54 users found this review helpful. Did you? Yes No

Report violations of our review guidelines.

Merchants and Prices (2 Merchants)

Buy & Download

Merchant	Merchant Rating	Price	Buy & Download
CNET Download.com	Rating not available	Free	
Tucows		Free	Free Download

Search. Windows Marketplace Go

List your merchandise here Help & FAQ Hardware Compatibility List Resource Center Windows.com Microsoft Update

 Site map Terms of use Privacy statement Disclaimer Team blog About Windows Marketplace

Search

Differences between the 32-bit and 64-bit versions of Internet Explorer that are included in the x64-based versions of Windows Server 2003 and in Windows XP Professional x64 Edition

On This Page

Article ID : 896457
Last Review : October 11, 2007
Revision : 2.6

SUMMARY

The x64-based versions of Microsoft Windows Server 2003 and Microsoft Windows XP Professional x64 Edition include a 32-bit version and a 64-bit version of Microsoft Internet Explorer. Both versions of Internet Explorer are included to increase compatibility with existing programs and Web sites. The 32-bit version of Internet Explorer can host only native 32-bit ActiveX controls and other 32-bit Web page objects. The 64-bit version of Internet Explorer can host only native 64-bit ActiveX controls and other 64-bit Web page objects.

This article describes the differences between the 32-bit and 64-bit versions of Internet Explorer that are included in the x64-based versions of Windows.

MORE INFORMATION

If you experience problems when you use the 64-bit version of Internet Explorer, try to view the same Web site by using the 32-bit version of Internet Explorer. If the problem occurs in both versions of Internet Explorer, you must determine whether the problem is caused by a Web object or if the problem is caused by Internet Explorer.

For more information, click the following article number to view the article in the Microsoft Knowledge Base:

154036 (http://support.microsoft.com/kb/154036/) How to disable active content in Internet Explorer

Both versions of Internet Explorer share the default home page, the **Favorites** menu, and the Temporary Internet Files folder. By default, the 32-bit version of Internet Explorer runs when you start the browser from any one of the following locations:

- The Quick Launch toolbar
- The desktop
- The **Start** menu
- A hyperlink or a file type that is associated with Internet Explorer

To start the 64-bit version of Internet Explorer, click **Start**, and then click **Internet Explorer (64-bit)**. You can also add desktop and Quick Launch icons to start the 64-bit version.

To determine the version of Internet Explorer that you are running, click **About Internet Explorer** on the **Help** menu of Internet Explorer. If you are running the 64-bit version of Internet Explorer, the **About Internet Explorer** window includes a 64-bit descriptor label. The 32-bit version of Internet Explorer does not show a bit-level descriptor.

Interopability with existing technologies

The differences between the 32-bit and 64-bit versions of Internet Explorer can affect toolbar add-ins and Microsoft ActiveX controls. For example, 32-bit toolbar add-ins do not work in the 64-bit version of Internet Explorer. These add-ins include the MSN toolbar, the AOL toolbar, the eBay toolbar, and the Google toolbar. You can install 32-bit toolbars in the 32-bit version of Internet Explorer. However, the 32-bit toolbars will not appear in the 64-bit version of Internet Explorer.

For example, when you start the Setup program for the 32-bit Google toolbar in the 64-bit version of Internet Explorer, the Google Setup program closes the 64-bit Internet Explorer window. The Google Setup program opens a new 32-bit Internet Explorer window when setup is completed. The Google toolbar appears in the 32-bit version of Internet Explorer, and the toolbar works correctly. The 32-bit Google toolbar is not installed in the 64-bit version of Internet Explorer.

By design, the 64-bit version of Internet Explorer cannot host 32-bit ActiveX controls. Additionally, the 32-bit version of Internet Explorer cannot host 64-bit ActiveX controls.

Default settings and storage

Both versions of Internet Explorer use the following folders:

- Temporary Internet Files are stored in the \Documents and Settings*profile*\Local Settings\Temporary Internet Files folder.
- The history list is stored in the \Documents and Settings*profile*\Local Settings\History folder.
- The favorites list is stored in the \Documents and Settings*profile*\Favorites folder.

Note *profile* is the folder that contains your account folders.

Additionally, both versions of Internet Explorer use the same default home page, security settings, and pop-up blockers.

You can open a Web page that was saved or copied from either the 32-bit version or the 64-bit version of System Monitor in both versions of Internet Explorer.

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, regarding the performance or reliability of these products.

For information about how to contact your vendor, click the appropriate article number in the following list to view the article in the Microsoft Knowledge Base:

65416 (http://support.microsoft.com/kb/65416/) Hardware and software vendor contact information, A-K

60781 (http://support.microsoft.com/kb/60781/) Hardware and software vendor contact information, L-P

60782 (http://support.microsoft.com/kb/60782/) Hardware and software vendor contact information, Q-Z

Technical support for Windows x64 editions

Your hardware manufacturer provides technical support and assistance for Microsoft Windows x64 editions. Your hardware manufacturer provides support because a Windows x64 edition was included with your hardware. Your hardware manufacturer might have customized the Windows x64 edition installation with unique components. Unique components might include specific device drivers or might include optional settings to maximize the performance of the hardware. Microsoft will provide reasonable-effort assistance if you need technical help with your Windows x64 edition. However, you might have to contact your manufacturer directly. Your manufacturer is best qualified to support the software that your manufacturer installed on the hardware.

For product information about Microsoft Windows XP Professional x64 Edition, visit the following Microsoft Web site:

http://www.microsoft.com/windowsxp/64bit/default.mspx
(http://www.microsoft.com/windowsxp/64bit/default.mspx)

For product information about Microsoft Windows Server 2003 x64 editions, visit the following Microsoft Web site:

http://www.microsoft.com/windowsserver2003/64bit/x64/editions.mspx
(http://www.microsoft.com/windowsserver2003/64bit/x64/editions.mspx)

APPLIES TO

- Microsoft Windows Server 2003, Enterprise x64 Edition
- Microsoft Windows Server 2003, Standard x64 Edition
- Microsoft Windows Server 2003, Datacenter x64 Edition
- Microsoft Windows XP Professional x64 Edition

Keywords: kbtshoot kbinterop KB896457

How to Protect Your Computer from Spyware and Adware

Published: April 20, 2004
By Jerry Honeycutt

As if spam, viruses, and worms aren't bad enough. Adware and spyware are here to sap the remaining life out of your productivity and privacy. Cookies are harmless in comparison!

Adware is software that displays advertisements on your computer. These are ads that inexplicably pop up on your display screen, even if you're not browsing the Internet. Some companies provide "free" software in exchange for advertising on your display. It's how they make their money.

Spyware is software that sends your personal information to a third party without your permission or knowledge. This can include information about Web sites you visit or something more sensitive like your user name and password. Unscrupulous companies often use this data to send you unsolicited targeted advertisements.

I've noticed more postings in the Microsoft Windows XP newsgroups about these threats. Many of the postings ask how they can tell if they have spyware on their systems and how to remove spyware if they find it. A small handful asks how to fix problems left over after removing spyware. I'm glad to see a lot of the advice offered from other enthusiasts and I'm going to share some of that advice with you in this month's column.

Related Links

- Discuss this column in the Windows XP Security and Administration Newsgroup
- Windows XP General Newsgroup
- Spyware and Deceptive Software
- McAfee Security
- Norton Antivirus
- Trend Micro
- Google Groups
- Google Toolbar
- Lavasoft
- SpywareInfo
- Spyware Protection and Removal
- Previous columns by Jerry Honeycutt

Windows Media Player 9 Series Questions

Now I know that a small number of you think of Windows XP, Windows Media Player, or Windows Messenger as spyware. The reason I disagree is that Microsoft provides a good combination of privacy notice and choice to users regarding the use of Web services and the sharing of information. For example, when you first run Windows Media Player 9 Series, you're given a chance to review the privacy options and make changes as you see fit. To further preserve your privacy, the default value of the player ID is set to "anonymous."

↑ Top of page

Is Your PC Affected by Spyware?

The main problem that most people notice with either kind of program is that they cause performance issues with their computers. For example, Internet Explorer might not work properly any more, your computer might hang more frequently, or your computer might slow down significantly. Removing spyware successfully is difficult enough to make preventing it in the first place a priority.

Unauthorized adware and spyware usually install on your computer covertly by using one of two methods:

- Tricking you into clicking a link that installs it. Links to spyware can be deceptive. For example, a Web site that's trying to push spyware onto your computer might open a window that looks like a Windows dialog box, and then trick you by installing when you click a Cancel button to close the dialog box. Sometimes, spyware pushers will put a fake title bar in an empty window, and then install spyware when you try closing the window.
- Installing freeware that includes it. For example, you might install a free file-sharing program that surreptitiously installs spyware on your computer. File-sharing programs can be a major conveyor of adware.

Once installed, spyware can transmit your personal information and download advertisements 24 hours a day. It can also hijack your browser settings, such as your home page or search page.

↑ Top of page

Protect against Spyware and Adware

Without help, you have no way to prevent adware or spyware. Old antivirus programs don't even prevent adware, since they didn't consider them viruses or worms. First, you usually give permission to install adware, although you do so unwittingly because adware and spyware pushers are deceptive. Second, adware doesn't behave like a typical virus or worm. They don't usually do actual damage to your computer, other than wrecking its performance, and they don't spread themselves using your address book. (Although some kinds of adware can break your anti-spyware tools.)

Things are changing for the better, though. Most popular antivirus products now include adware and spyware scanning. For example, the latest versions of McAfee VirusScan, Norton AntiVirus 2004, and Trend Micro PC-Cillin 2004 now scan for some adware and spyware.

Also, some Internet service providers (ISPs) are introducing protection from adware and spyware. For example, America Online (AOL) announced in January spyware protection as an enhancement for AOL 9.0 Optimized. EarthLink also provides adware and spyware protection through the latest version of its software. Of course, to take advantage of the built-in protection that antivirus products and ISPs provide, you have to update to the latest versions, and keep the anti-spyware/adware signatures current.

↑ Top of page

Prevent Unwanted Installation

Companies pushing adware and spyware are relying on two things: your desire for free software and your gullibility. I've had two friends bring me their computers after they were seriously infected with adware. In one case, the culprit was my friend's craving for free file-sharing software. His desktop was a mess with countless icons for programs that he downloaded from the Internet. I was aghast. What he didn't realize is that he gave implicit permission to install adware. He knows better now.

My other friend isn't a freeware glutton. Instead, she has a habit of clicking the Yes or OK buttons on every dialog box she sees. Even suspicious-looking dialog boxes that don't pass close scrutiny. Of course, when a dialog box pops up asking if it's OK to install a new program, she clicks the Yes button.

The lesson that you can learn from my friends will help you prevent the installation of most adware and spyware:

- **Make sure the programs you install don't contain adware.** Many freeware programs do include adware. It's how the publishers make their money. If you're not sure, read the license agreement carefully (these are usually shown directly or through links as part of the installation process). Also, check the publisher's Web site very carefully. If you're still not sure, search Google Groups for the name of the program and the keywords *adware* or *spyware*. If you don't find any postings about it, then you're probably OK.
- **Install a pop-up blocker to prevent adware and spyware pop-up windows.** Much spyware installs after you click a deceptive link in a pop-up browser window. Install a pop-up blocker, and you won't even be tempted to click those links. My two favorite pop-up blockers are completely free. The first is the new MSN Toolbar. The second is the Google Toolbar. Pop-up windows are annoying time wasters anyway, so you'll thank yourself later. If you're a Windows XP user, look for a service pack this summer (Service Pack 2) that will include a number of great security features, as well as a pop-up blocker for Internet Explorer.
- **Don't unwittingly install adware or software.** If you do click what seems like a innocuous link, and then you see a dialog box similar to the one shown in Figure 1, *don't click the Yes button to install the software*. In this example, I was expecting to install a program from Microsoft so I feel safe. If in doubt, however, do not proceed. This dialog box is your last line of defense, and you should only install programs from the Internet that you chose to install. This is akin to giving someone your credit card number who calls you at home. It's a different story if you called them. Installing Windows XP SP 2 (when it's available later this summer) will also provide some help by suppressing unsolicited downloads of ActiveX controls (a popular vehicle for spyware).

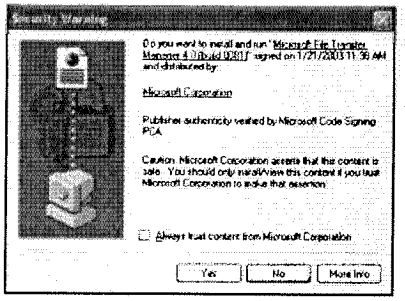


Figure 1: Only click Yes if you trust the publisher and want the software.

Spyware scanners and some virus scanners with spyware signatures can help combat spyware. However, the best strategy is to be discriminating about what you choose to download and install.

↑ Top of page

Check Your Computer

If you're even thinking about scanning your computer for adware and spyware, then you're probably experiencing some of the symptoms I described earlier in this article. Those include instability, performance problems, or possibly a hijacked Web browser.

There is software specifically designed for detecting spyware and adware, and helping you remove it. The one with which I'm most familiar is Ad-aware from Lavasoft. This is the program that I recommend to most of my friends. A freeware version is available for use by individuals at home. A commercial version is also available for use in corporate environments. A program like Ad-aware finds adware and spyware on your computer and then removes them.

You can find more adware and spyware removal tools at the Spyware Protection and Removal guide. This Web page includes links to popular spyware removal programs, as well as a number of useful articles. If you're not going to use a popular program like Ad-aware, however, search Google Groups for the name of the program you do choose. Some spyware removal software can cause as many problems as it fixes, and you want to find out about these problems before using unproven software.

Tip: Malke, a Microsoft MVP, offers this tip in the Windows XP Newsgroups: "It's best to run antivirus and spyware removal tools in Safe Mode." This is because removal tools sometimes can't remove spyware from your computer while it's running.

↑ Top of page

Get More Help

The best place to ask questions about adware if you suspect your computer is infested is in the newsgroups. Specifically, the Windows XP Security and Administration and the Windows XP General newsgroups tend to be where most users post and answer these types of questions. Rather than wait for an answer to your question, however, I suggest that you search the Windows XP newsgroups at Google Groups.

When you do post your questions, make sure you give a thorough description of the symptoms you're experiencing. The more information you give, the easier it will be for other people to help you. For example, you'll want to describe your hardware configuration as much as possible. It is also important to describe any software that you've recently installed, since unwanted software often comes bundled with other applications. Be sure to describe any pop-up windows that have suddenly started appearing on your desktop and, if possible, include a screenshot of them.

Remember not to post anything in a newsgroup that you'd have a problem with millions of people seeing—particularly malicious people who would take advantage of personal information. So don't provide account names, IP addresses, or passwords.

Even after posting your question, don't be disappointed if someone tells you to run a scanner like Ad-aware. Generally, if you suspect that your computer is infested with adware or spyware, this is the best advice.

Expert Zone Columnist Jerry Honeycutt is a writer, speaker, and technologist who has written over 25 books, including Microsoft Windows XP Registry Guide *(Microsoft Press, 2002). He frequently writes about customizing and deploying Windows XP.*

↑ Top of page
Printer-Friendly Version | Send This Page | Add to Favorites

Manage Your Profile | Contact Us | Free Newsletter | RSS Feeds

Search

"Iexplorer.exe has encountered a problem and needs to close" error message when you try to view a timesheet in Microsoft Project Web Access

On This Page

Article ID : 875344
Last Review : January 18, 2007
Revision : 1.1

SYMPTOMS

When you try to view a timesheet in Microsoft Project Web Access, Microsoft Internet Explorer quits, and you receive the following error message:

Iexplorer.exe has encountered a problem and needs to close

When you click **more information**, you receive the following error message where *x.x.xxxx.xxxx* is the version of Internet Explorer, and *yyyyyyy* is the offset:

AppName: iexplore.exe
AppVer: *x.x.xxxx.xxxx*
ModName: pjgrid.ocx
ModVer: 9.0.2000.224
Offset: *yyyyyyy*

CAUSE

This problem may occur if the Google Toolbar for Internet Explorer is installed on your computer.

WORKAROUND

To work around this problem, use one of the following methods.

Method 1

Remove the Google Toolbar for Internet Explorer. To do this, follow these steps:

1. Click **Start**, click **Control Panel**, and then click **Add or Remove Programs**.
2. Click **Google Toolbar for Internet Explorer**, and then click **Change/Remove**.

Method 2

Disable third-party browser extensions in Internet Explorer. To do this, follow these steps:

1. Start Internet Explorer.
2. On the **Tools** menu, click **Internet Options**, and then click the **Advanced** tab.
3. On the **Advanced** tab, click to clear the **Enable third-party browser extensions (requires restart)** check box, and then click **OK**.

Note Both methods require that you restart Internet Explorer.

MORE INFORMATION

You may not experience this problem with a later version of the Google Toolbar. For more information about the Google Toolbar for Internet Explorer, visit the following Google Web site:

http://toolbar.google.com (http://toolbar.google.com)

Microsoft provides third-party contact information to help you find technical support. This contact information may change without notice. Microsoft does not guarantee the accuracy of this third-party contact information.
The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, regarding the performance or reliability of these products.

APPLIES TO

- Microsoft Project Server 2002
- Microsoft Project 2002 Professional Edition
- Microsoft Project 2000 Standard Edition
- Microsoft Office Project Web Access

Keywords: kb3rdparty kbtshoot kbprb KB875344

Snip IT Add-On for Internet Explorer

Microsoft

Brief Description

Use this add-on to send text selected in Internet Explorer as e-mail.

On This Page

Download

Quick Details

File Name:	SnipIT.msi
Version:	1.0.1
Date Published:	2/5/2007
Language:	English
Download Size:	184 KB
Estimated Download Time:	Dial-up (56K) 1 min

Overview

SnipIT can be used to e-mail the selected text in Internet Explorer using the user's e-mail client of choice. The add-on currently supports Microsoft Outlook, Microsoft Hotmail, Yahoo Mail, Gmail and default installed MailTo client in Internet Explorer as e-mail client options. Yahoo Mail Beta is not currently supported.

Top of page

System Requirements

- **Supported Operating Systems:** Windows Vista; Windows XP

Microsoft Internet Explorer 6.0 and later

Top of page

Instructions

The program can be run directly from the Web site by clicking 'Run' when prompted. If you choose to save the program, run the program from the location to which it was saved.

Top of page

Additional Information

- Running this program does not send any information to Microsoft.
- Snip IT addon functionality will only be available on Internet Explorer pages which are opened later to installing this addon.
- You can remove this program at any time.
- This tool does not support Yahoo Mail (Beta).
- Older versions of Google toolbar may prevent this Addon from functioning properly. Install latest version of Google toolbar (if required).

Top of page

Related Resources

1. Internet Explorer Add-Ons on Windows Marketplace
2. Internet Explorer Home Page
3. Internet Explorer Community
4. Most Popular Internet Explorer Downloads

Top of page

What Others Are Downloading

Others who downloaded **Snip IT Add-On for Internet Explorer** also downloaded:

1. Windows Internet Explorer 7 for Windows XP SP2
2. Live Search Maps Add-In for Microsoft Office Outlook
3. Virtual Earth 3D (Beta)
4. Clear Cache Feature for Internet Explorer
5. Update for Internet Explorer 7 for Windows XP (KB928089)

Top of page

Search

Internet Explorer 6 software update and its effect on ActiveX controls

NOTICE

The update that is described in this article is included in the most current cumulative security update for Internet Explorer. To install the most current update, install all important or high-priority updates. To do this, visit the Windows Update Web site:

Article ID : 912945
Last Review : December 27, 2007
Revision : 14.2

http://windowsupdate.microsoft.com (http://windowsupdate.microsoft.com)

For more technical information about the most current cumulative security update for Internet Explorer, visit the following Microsoft Web site:

http://www.microsoft.com/technet/security/current.aspx
(http://www.microsoft.com/technet/security/current.aspx)

This article is intended to notify IT professionals and developers about a change in the way Internet Explorer handles ActiveX controls. Home users can visit the following Microsoft Web site for information about how Internet Explorer handles ActiveX controls when this update is installed:

http://www.microsoft.com/windows/ie/ie6/using/techinfo/activexupdate.mspx (http://www.microsoft.com/windows/ie/ie6/using/techinfo/activexupdate.mspx)

Note If you are experiencing problems with an ActiveX control on a Web site that you trust, add the Web site to the Trusted Sites zone in Internet Explorer. To do this, follow these steps in Internet Explorer:

1. On the **Tools** menu, click **Internet Options**.
2. On the **Security** tab, click **Trusted Sites**, and then click **Sites**.
3. Under **Add this website to the zone**, type the URL of the Web site that you want to add.
4. Click **Add**, and then click **Close**.

If the problem persists, visit one of the following Microsoft Web sites:

- **Internet Explorer 6 Solution Center**
  http://support.microsoft.com/ph/2073 (http://support.microsoft.com/ph/2073)
- **Internet Explorer 7 Solution Center**
  http://support.microsoft.com/ph/8722 (http://support.microsoft.com/ph/8722)

On This Page

INTRODUCTION

Microsoft has released a software update to Microsoft Internet Explorer 6 for Windows XP Service Pack 2 (SP2) and for Windows Server 2003 Service Pack 1 (SP1). This update changes how Internet Explorer handles some Web pages that use ActiveX controls and Java add-ins. Examples of ActiveX controls include the following:

Adobe Reader
Apple QuickTime Player
Macromedia Flash Player
Microsoft Windows Media Player
Real Networks RealPlayer
Sun Java Virtual Machine

After you install this update, you cannot interact with ActiveX controls from certain Web pages until these controls are enabled. To enable an ActiveX control, manually click the control. There are also techniques that Web developers can use to update their Web pages. For more information about these techniques, visit the following MSDN Web site:

http://msdn2.microsoft.com/en-us/library/ms537508.aspx (http://msdn2.microsoft.com/en-us/library/ms537508.aspx)

When you install this update, the Plugin.ocx binary is completely removed from Windows Server 2003 and Windows XP. The Plugin.ocx binary is a private component of Internet Explorer without any public interfaces. Plugin.ocx is used to host Netscape plug-ins as ActiveX controls. The functionality of Plugin.ocx was disabled in 2003 in Windows Server 2003 and Windows XP for security reasons. This update removes the nonfunctional Plugin.ocx code completely.

As part of this Internet Explorer update, Microsoft will release updates to the current versions of Windows XP and of Windows Server 2003. All client operating systems will be updated. These client operating systems include the following:

- Windows XP Starter Edition
- Windows XP Home Edition
- Windows XP Professional Edition
- Windows XP Tablet PC Edition
- Windows XP Media Center Edition
- Windows XP Professional for Embedded Systems

Currently, Microsoft has not released updates for earlier versions of Internet Explorer, Windows Server 2003, or Windows XP. However, Microsoft may release updates for these earlier versions in the future. More information about release schedules will be posted in this article when the information becomes available.

The following files are available for download from the Microsoft Download Center:

Update for Internet Explorer for Windows Server 2003, 32-bit x86-based versions with Service Pack 1:

Download the 912945 package now. (http://www.microsoft.com/downloads/details.aspx?FamilyId=4196F9BE-9022-4AD6-9F4B-5359318FA8BF&displaylang=en)

Update for Internet Explorer for Windows Server 2003, 64-bit Itanium-based versions with Service Pack 1:

Download the 912945 package now. (http://www.microsoft.com/downloads/details.aspx?FamilyId=F8D62434-5014-4F8E-8200-C25127AC3207&displaylang=en)

Update for Internet Explorer for Windows Server 2003, 64-bit x64-based versions:

Download the 912945 package now. (http://www.microsoft.com/downloads/details.aspx?FamilyId=6D56C625-FAD1-412D-A740-9C4DBF804820&displaylang=en)

Update for Internet Explorer for Windows XP Professional x64 Edition:

Download the 912945 package now. (http://www.microsoft.com/downloads/details.aspx?FamilyId=37431BBA-2D8E-48AB-8C9F-D5F5B2EA7BE7&displaylang=en)

Update for Internet Explorer for Windows XP with Service Pack 2:

Download the 912945 package now. (http://www.microsoft.com/downloads/details.aspx?FamilyID=999d37c3-4013-48de-b950-ee01256aaa92&displaylang=en)

For more information about how to download Microsoft support files, click the following article number to view the article in the Microsoft Knowledge Base:

119591 (http://support.microsoft.com/kb/119591/) How to obtain Microsoft support files from online services

Microsoft scanned this file for viruses. Microsoft used the most current virus-detection software that was available on the date that the file was posted. The file is stored on security-enhanced servers that help prevent any unauthorized changes to the file.

MORE INFORMATION

Known issues

- **Internet Explorer ActiveX update that is contained in security update 912812 is disabled**

  After you deploy update 912945 for Internet Explorer, the behavior of the Internet Explorer ActiveX update that is contained in security update 912812 is disabled. The security fixes that are contained in security update 912812 are still present and will still function. Only the Internet Explorer ActiveX update behavior is disabled.

- **Web page rendering issues within Internet Explorer**

  This issue occurs if Mshtml.dll is ever reregistered after you install this cumulative update. This issue is resolved in security update 912812 for Internet Explorer.

  912812 (http://support.microsoft.com/kb/912812/) MS06-013: Cumulative security update for Internet Explorer

  We recommend that you install security update 912812 for Internet Explorer instead of installing update 912945 that is described in this article.

- **Initial logon dialog boxes may reappear and reset to default configurations**

  This issue occurs if you deploy the hotfix version of this software update on 64-bit systems, such as an Itanium-based or x64-based version of Windows Server 2003 or an x64-based version of Windows XP with their respective service packs. In this case, the initial logon dialog boxes may appear for applications and for Windows components. Additionally, some settings reset to default. This behavior may cause the following issues:

- Applications ask users to opt in to privacy features.
- Default settings for Internet Explorer favorites are reset.
- Internet Explorer security zones are reset to default settings.
- Internet Explorer advanced settings are reset to default settings.
- Initial Windows Media Player dialog boxes appear.

This issue is resolved in security update 912812 for Internet Explorer.

- **Google Toolbar**

  You may experience an access violation in the Google Toolbar when you close a window that contains an inactive ActiveX control. Microsoft and Google technical teams have been working together to address this issue. Google is expected to fix this problem by using its automatic "servicing mechanism" for Google Toolbar users. This problem affects Google Toolbar versions before version 3.0.129.2. Visit the following Google Web site to download the latest version:

  http://toolbar.google.com (http://toolbar.google.com)

- **External script technique does not work when the "Disable Script Debugging (Internet Explorer)" check box is cleared**

  This issue is resolved in security update 912812 for Internet Explorer.

- **ActiveX controls that use Java Platform, Standard Edition 1.3 or 1.4**

  After you click an ActiveX add-in control in a program that runs the add-in control by using Java Platform, Standard Edition (J2SE) 1.3 or J2SE 1.4, the focus does not move to the add-in control. You must click the control again to establish focus. The focus behavior works correctly in J2SE 1.5. To obtain the latest version of J2SE, visit the following Sun Microsystems, Inc. Web site:

  http://java.sun.com/j2se (http://java.sun.com/j2se)

- **Unable to use the /integrate switch to update Windows installation source files**

  Administrators cannot use the **/integrate** switch to update Windows installation source files with this update. This is expected to be fixed in the next update for Windows Server 2003 and Windows XP. To work around this issue, use the Sysprep tool. For more information, visit the following Microsoft Web site:

  http://technet.microsoft.com/en-us/library/bb457067.aspx (http://technet.microsoft.com/en-us/library/bb457067.aspx)

  This issue is resolved in security update 912812 for Internet Explorer.

- **Siebel programs that use ActiveX controls**

  Software update 912945 affects all Siebel 7 High Interactive clients. After you apply this update, you must click several times to interact with the Siebel program, one time for each ActiveX control in the program. Siebel is working with Microsoft to determine a solution. A Siebel product update is expected to be released some time in the spring of 2006. For more information about Siebel product updates, visit the following Siebel Support Web site:

  https://ebusiness.siebel.com/supportweb/ (https://ebusiness.siebel.com/supportweb/)

- **Substring match opt-in process names**

  By default, certain applications have process names that are a subset of applications that are opted in to the new ActiveX behavior. An example would be APExplorer.exe, where a substring of this name is Explorer.exe. Therefore, Explorer.exe is opted in to the new ActiveX behavior. Such applications exhibit the new ActiveX behavior.

  This issue is resolved in security update 912812 for Internet Explorer.

- **MFC controls leave a permanent window**

  In certain cases, when moving away from a page that has an MFC ActiveX control, the control still appears in the new window. This issue is resolved in security update 916281 for Internet Explorer.

- **Visual Basic controls do not appear**

  In certain cases, controls that are created in Visual Basic that are displayed with display and visibility CSS attributes may not appear. This issue is resolved in security update 916281 for Internet Explorer.

- **Security warning message when you try to open a PDF document from a secure (https://) Web page**

  When you try to open a PDF document from a secure (https://) Web page, you incorrectly receive a security warning message for mixed content. This issue is resolved in security update 916281 for Internet Explorer.

For recommended techniques to make sure that ActiveX controls function without user interaction, visit the following MSDN Web site:

http://msdn2.microsoft.com/en-us/library/ms537508.aspx (http://msdn2.microsoft.com/en-us/library/ms537508.aspx)

The following issues occur on Web sites that do not use the recommended techniques.

Note All these issues are resolved by using the techniques that are described on the MSDN Web site.

- **Scrolling**
  When you use the mouse wheel to scroll through a page that contains an interactive control, the control may not be displayed correctly. This issue is resolved in security update 912812 for Internet Explorer.
- **Abstract Window Toolkit**
  Access violations have been reported with Java programs that use Abstract Window Toolkit (AWT) classes in the user interface. This issue is resolved in security update 912812 for Internet Explorer.
- **Transparent Flash**
  A full-page ad disappears. However, the focus rectangle remains. In this situation, the control is still there. However, it is transparent. Therefore, the associated overlay window remains on the page.
- **DHTML menus**
  When a DHTML menu is expanded, the menu may appear on top of an ActiveX control. If you click the menu in this situation, you enable the control instead of gaining access to the DHTML menu. The overlay window has the highest z-order. Therefore, this window receives the mouse-click message.
- **Controls that prompt before they are loaded**
  When certain controls are loaded on a Web page, the controls are not correctly masked by the functionality of this update. These controls include controls that are used in Macromedia Shockwave Director, in QuickTime Player, and in Virtools Web Player. When Windows determines that a control is inactive, the system prompts the user before the control is loaded.
- **CSS attributes on controls**
  Controls that are hidden or that have a display-mode setting of **None**, but that do have size dimensions, display the focus rectangle when you move the pointer over them.
  This issue is resolved in security update 916281 for Internet Explorer.

Technical support for x64-based versions of Microsoft Windows

If your hardware came with a Microsoft Windows x64 edition already installed, your hardware manufacturer provides technical support and assistance for the Windows x64 edition. In this case, your hardware manufacturer provides support because a Windows x64 edition was included with your hardware. Your hardware manufacturer might have customized the Windows x64 edition installation by using unique components. Unique components might include specific device drivers or might include optional settings to maximize the performance of the hardware. Microsoft will provide reasonable-effort assistance if you must have technical help with a Windows x64 edition. However, you might have to contact your manufacturer directly. Your manufacturer is best qualified to support the software that your manufacturer installed on the hardware. If you purchased a Windows x64 edition such as a Windows Server 2003 x64 edition separately, contact Microsoft for technical support.

For product information about Windows XP Professional x64 Edition, visit the following Microsoft Web site:

http://www.microsoft.com/windowsxp/64bit/default.mspx
(http://www.microsoft.com/windowsxp/64bit/default.mspx)

For product information about x64-based versions of Windows Server 2003, visit the following Microsoft Web site:

http://www.microsoft.com/windowsserver2003/64bit/x64/editions.mspx
(http://www.microsoft.com/windowsserver2003/64bit/x64/editions.mspx)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

APPLIES TO

- Microsoft Internet Explorer 6.0, when used with:
  Microsoft Windows Server 2003, Standard x64 Edition
  Microsoft Windows Server 2003, Datacenter x64 Edition
  Microsoft Windows Server 2003, Enterprise x64 Edition
  Microsoft Windows Server 2003 Service Pack 1
  Microsoft Windows XP Professional x64 Edition
  Microsoft Windows XP Service Pack 2

Keywords: kbresolve kbwebbrowser kbactivexscript kbonline kbexpertiseadvanced kbhowto kbfaq KB912945

EXHIBIT K-1

LIBRARY OF CONGRESS ONLINE CATALOG

DATABASE: Library of Congress Online Catalog
YOU SEARCHED: Author/Creator Browse = graham, ian s.
SEARCH RESULTS: Displaying 6 of 8.

◀ Previous Next ▶

Brief Record | Subjects/Content | Full Record | MARC Tags

The HTML sourcebook / Ian S. Graham.

LC Control No.: 94047369
LCCN Permalink: http://lccn.loc.gov/94047369
Type of Material: Book (Print, Microform, Electronic, etc.)
Personal Name: Graham, Ian S., 1955-
Main Title: The HTML sourcebook / Ian S. Graham.
Published/Created: New York : Wiley, c1995.
Description: xvi, 416 p. : ill. ; 24 cm.
ISBN: 0471118494 (pbk. : acid-free paper)

CALL NUMBER: QA76.76.H94 G73 1995 FT MEADE
Copy 1
-- **Request in:** Main or Science/Business Reading Rms - STORED OFFSITE
-- **Status:** Not Charged

CALL NUMBER: QA76.76.H94 G73 1995 FT MEADE
Copy 2
-- **Request in:** Main or Science/Business Reading Rms - STORED OFFSITE
-- **Status:** Not Charged

◀ Previous Next ▶

EXHIBIT K-2

```
<!-- Start of HTML header portion -->
<link rel="stylesheet" type="text/css" href="/css/basic.css" />
<link rel="stylesheet" type="text/css" href="/css/timer.css" />
<script type="text/javascript" src="/js/valid.js"></script>
<script type="text/javascript" src="/js/timer.js"></script>
<script type="text/javascript" src="/js/zdragdroplib.js"></script>
<table width="98%" border="0" cellspacing="0" cellpadding="0" align=center>
<tr>
<td colspan=5 align=left><table width="100%" cellpaddiing=0 cellspacing=0 border=0><tr>
<td width="53%"><font face="Arial, Helvetica, sans-serif" size="-1"><a href="http://www.loc.gov/">The
Library of Congress</a></font></td>
<td align=right width="47%"><font face="Arial, Helvetica, sans-serif" size="-1">&gt;&gt;
<a href="http://authorities.loc.gov/"> Go to Library of Congress Authorities</a></font></td>
</tr></table></td>
</tr>
<tr height=55>

<td width=57><img src="/images/banner-rose-left.gif" width="57" height="55" alt=""></td>

<td width="50%" background="/images/banner-back.gif"> </td>

<td width=488><A HREF="/webvoy.htm"><img src="/images/banner-center.gif" width="488" height="55" alt="Start Over" border="0"></a></td>

<td width="50%" background="/images/banner-back.gif"> </td>

<td width=60><img src="/images/banner-rose-right.gif" width="60" height="55" alt=""></td>
</tr>
</table><!-- End of HTML header portion -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
<meta http-equiv="Content-Type" Content="text/html;charset=UTF-8">
<TITLE>Brief Record (Library of Congress Online Catalog)</TITLE>
</HEAD>
<BODY  bgcolor="#ffffff" text="#000000" link="#006666" vlink="#993300"><CENTER>
<FONT face=Default size=3><A HREF="/help/disphlp1.htm"><IMG ALIGN="MIDDLE" BORDER=0 ALT="Help" SRC="/images/UpHelp.gif"></A><A HREF="/cgi-bin/Pwebrecon.cg
><IMG ALIGN="MIDDLE" BORDER=0 ALT="Headings List" SRC="/images/UpHeadings.gif"></A><A HREF="/cgi-bin/Pwebrecon.cgi?ti=1,0&Search%5FArg=graham%2C%20ian%20s
><IMG ALIGN="MIDDLE" BORDER=0 ALT="Titles List" SRC="/images/UpTitles.gif"></A><INPUT TYPE=HIDDEN NAME=BCKT VALUE=><A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=RE
<INPUT TYPE=HIDDEN NAME=Search_Arg VALUE="graham, ian s."><INPUT TYPE=HIDDEN NAME=Search_Code VALUE="NAME_"><INPUT TYPE=HIDDEN NAME=SL VALUE=""><INPUT TYP
<TR>
<TD><font size="-1" color="#006666"><b>DATABASE:</b></font> Library of Congress Online Catalog</TD>
<BR>
</TR>
<TR>
<TD ALIGN=LEFT><font size="-1" color="#336666"><b>YOU SEARCHED</b></font>: Author/Creator Browse = graham, ian s.</TD>
</TR>
<TR>
<TD ALIGN=LEFT><font size="-1" color="#336666"><b>SEARCH RESULTS</b></font>: Displaying 6 of 8.</TD>
</TR>
</TABLE>
<CENTER>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=5&ti=1,5&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Previous Screen or Record" SRC="/images/s-prev.gif"></A>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=7&ti=1,7&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Next Screen or Record" SRC="/images/s-next.gif"></A>
</CENTER>
<CENTER><IMG SRC="/images/view1a.gif" ALIGN=MIDDLE BORDER=0 ALT="Brief Record"><A HREF="/cgi-bin/Pwebrecon.cgi?v2=6&ti=1,6&SEQ=20080228120449&Search%5FArg
><IMG SRC="/images/view2.gif" ALIGN=MIDDLE BORDER=0 ALT="Subjects/Content"></A><A HREF="/cgi-bin/Pwebrecon.cgi?v3=6&ti=1,6&SEQ=20080228120449&Search%5FArg
><IMG SRC="/images/view3.gif" ALIGN=MIDDLE BORDER=0 ALT="Full Record"></A><A HREF="/cgi-bin/Pwebrecon.cgi?v4=6&ti=1,6&SEQ=20080228120449&Search%5FArg=grah
><IMG SRC="/images/view4.gif" ALIGN=MIDDLE BORDER=0 ALT="MARC Tags"></A></CENTER><P>
<CENTER><H3></H3></CENTER>
<CENTER>
<H4>
<I>
<STRONG><BDO dir="LTR">The HTML sourcebook / Ian S. Graham.</BDO></STRONG>
</I>
</H4>
</CENTER>
<INPUT TYPE=HIDDEN NAME=RID VALUE=2071355><TABLE>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>LC Control No.:</TH>
<TD dir="ltr">
    94047369 </TD></TR><TR><TH NOWRAP ALIGN=RIGHT VALIGN=TOP>LCCN Permalink:</TH>
<TD><a href="http://lccn.loc.gov/   94047369 " target="_blank">http://lccn.loc.gov/   94047369 </a></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Type of Material:</TH>
<TD>
Book (Print, Microform, Electronic, etc.)</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Personal Name:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=Author&SEQ=20080228120449&PID=3898&SA=Graham,+Ian+S.,+1955-"> Graham, Ian S., 1955- </A></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Main Title:</TH>
<TD dir="ltr">
 The HTML sourcebook / Ian S. Graham.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Published/Created:</TH>
<TD dir="ltr">
 New York : Wiley, c1995.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Description:</TH>
<TD dir="ltr">
 xvi, 416 p. : ill. ; 24 cm.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>ISBN:</TH>
<TD dir="ltr">
 0471118494 (pbk. : acid-free paper)</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
```

```
<TD></TD></TR>
<TR><TD COLSPAN=2><A name="D1"></A><HR WIDTH=150></TD></TR>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>CALL NUMBER:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=CallNumber&SEQ=20080228120449&PID=3898&SA=QA76.76.H94"> QA76.76.H94 G73 1995 FT MEADE</A></TD>
</TR>
<TR>
<TD></TD>
<TD dir="ltr">Copy 1</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Request in:</TH>
<TD>
Main or Science/Business Reading Rms - STORED OFFSITE</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
<TD></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Status:</TH>
<TD>
Not Charged</TD></TR>
<TR><TD COLSPAN=2><A name="D2"></A><HR WIDTH=150></TD></TR>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>CALL NUMBER:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=CallNumber&SEQ=20080228120449&PID=3898&SA=QA76.76.H94"> QA76.76.H94 G73 1995 FT MEADE</A></TD>
</TR>
<TR>
<TD></TD>
<TD dir="ltr">Copy 2</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Request in:</TH>
<TD>
Main or Science/Business Reading Rms - STORED OFFSITE</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
<TD></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Status:</TH>
<TD>
Not Charged</TABLE>
<BR><CENTER>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=5&ti=1,5&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Previous Screen or Record" SRC="/images/s-prev.gif"></A>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=7&ti=1,7&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Next Screen or Record" SRC="/images/s-next.gif"></A>
</CENTER>
<BR><INPUT TYPE=HIDDEN NAME=BIB VALUE=2071355><INPUT TYPE=HIDDEN NAME=PID VALUE="3898">
<INPUT TYPE=HIDDEN NAME=SEQ VALUE="20080228120449">
<CENTER>
<TABLE BORDER=1 BGCOLOR="#FFFFDD">
  <TR>
      <TH>Save, Print or Email Records (View <a href="/help/savemail.htm">Help</a>)</TH>
   </TR>
  <TR>
     <TD> Select Download Format:
        <SELECT NAME=RD>
            <OPTION VALUE=0>Text (Brief Information)</OPTION>
            <OPTION VALUE=1>Text (Full Information)</OPTION>
            <OPTION VALUE=2>MARC (non-Unicode/MARC-8)</OPTION>
            <OPTION VALUE=3>MARC (Unicode/UTF-8)</OPTION>
        </SELECT>
<INPUT TYPE="SUBMIT" NAME="SAVE" VALUE="Press to SAVE or PRINT" onClick="document.FAKEFORM.PS.value='1'">
</TD>  </TR>
  <TR>
               <TD><font size="-1" color="#336666"><b>Email Text (Full Info) to:</b></font><br>
                       <INPUT NAME="MAILADDY" MAXLENGTH=45 SIZE=32 onKeyPress="return submitenter(this,event)">
<SCRIPT  LANGUAGE="Javascript">
function submitenter(myfield,e){
        var keycode
        if(window.event) keycode = window.event.keyCode
        else if(e) keycode = e.which
   else return true
        if(keycode == 13){
var v1 = document.frm.MAILADDY.value;errorOccured = false;if(v1.length == 0){errorOccured = true; alert("No email address has been entered. Please enter a
document.frm.MAILADDY.focus();return true;}
if(!errorOccured){document.frm.EMAILADDRESS.value = "EMAIL"
document.frm.submit();return false;}}
else return true
}</SCRIPT>
<INPUT TYPE="button" NAME="EMAIL" VALUE="Press to SEND EMAIL" onClick=checkValues()>
<INPUT TYPE="HIDDEN" NAME="EMAILADDRESS" VALUE="None">
  </TR>
</TABLE></CENTER>
<SCRIPT  LANGUAGE="Javascript">
function checkValues(){var v1 = document.frm.MAILADDY.value;errorOccured = false;var email = /(\w+)@(\w+)/;
if(v1.length == 0){errorOccured = true; alert("No email address has been entered. Please enter an email address")
document.frm.MAILADDY.focus();}
else if(email.test(document.frm.MAILADDY.value)== false){
errorOccured = true; alert("Please enter a valid email")
document.frm.MAILADDY.focus();}
if(!errorOccured){document.frm.EMAILADDRESS.value = "EMAIL"
document.frm.submit()}}</SCRIPT>
</FORM><BR><FORM NAME="FAKEFORM">
<INPUT TYPE="HIDDEN" NAME="PS" VALUE='0'>
</FORM>
<CENTER>
<FONT face=Default size=3><A HREF="/help/disphlp1.htm"><font size="-1" face="Arial, Helvetica">Help</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi
><font size="-1" face="Arial, Helvetica">History</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?hd=0,0&PID=3898&HEADBROWSE=0&HIST=0&SEQ=2008022812
><font size="-1" face="Arial, Helvetica">Headings</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?ti=1,0&Search%5FArg=graham%2C%20ian%20s%2E&Search
A><font size="-1" face="Arial, Helvetica">Titles</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=REQUESTBIB&SEQ=20080228120449&PID=3898"
><font size="-1" face="Arial, Helvetica">Request</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=pbLogon&PID=3898"><font size="-1" face="Arial
><font size="-1" face="Arial, Helvetica">Exit</font></A></FONT></CENTER></BODY>
```

```
<!-- Start of HTML footer portion -->
<hr noshade size=1>
<table width="100%" border="0" cellspacing="6" cellpadding="4">
<tr valign="middle">
<td width="34%"><img src="/images/small-disc.jpg" alt="" width="90" height="87" hspace="2" vspace="0" align="left">
<strong><font face="Arial, Helvetica, sans-serif" size="-1">The Library
of Congress</font></strong><font face="Arial, Helvetica, sans-serif" size="-1"><br>
URL: <a href="http://www.loc.gov/">http://www.loc.gov/</a><br>
<i>Mailing Address: </i></font><font face="Arial, Helvetica, sans-serif" size="-1"><br>
101 Independence Ave, S.E.<br>
Washington, DC 20540</font></td>
<td width="36%" align="center" bgcolor="#FFFFdd"><font size="-1" face="Arial, Helvetica, sans-serif"><b>Catalog/authority
record errors?<br>
</b><a href="http://www.loc.gov/help/contact-libarch-report.html">Use our
Error Report Form</a><br>
<strong>Questions about searching?</strong><br>
<a href="http://www.loc.gov/rr/askalib/ask-digital.html">Ask a Librarian</a></font></td>
<td width="30%" align="right">
<p><font face="Arial, Helvetica, sans-serif" size="-1"><b>Library of Congress
Online Catalog</b><br>
<i>URL:</i> <a href="http://catalog.loc.gov/">http://catalog.loc.gov/</a><br>
<b>Library of Congress Authorities</b><br>
<i>URL:</i> <a href="http://authorities.loc.gov/">http://authorities.loc.gov/</a></font></p>
</td>
</tr>
</table>
<script type="text/javascript">
<!--//
document.write(getWarning());
//-->
</script>
<div id="warningDiv"></div>
<img src="/images/load.gif" onLoad="initTimer(); getElements();" />
<!-- End of HTML footer portion -->
</HTML>
```

EXHIBIT L-1

The Library of Congress >> Go to Library of Congress Authorities

LIBRARY OF CONGRESS ONLINE CATALOG

DATABASE: Library of Congress Online Catalog
YOU SEARCHED: Author/Creator Browse = graham, ian s.
SEARCH RESULTS: Displaying 6 of 8.

The HTML sourcebook / Ian S. Graham.

LC Control No.: 94047369
LCCN Permalink: http://lccn.loc.gov/94047369
Type of Material: Book (Print, Microform, Electronic, etc.)
Personal Name: Graham, Ian S., 1955-
Main Title: The HTML sourcebook / Ian S. Graham.
Published/Created: New York : Wiley, c1995.
Description: xvi, 416 p. : ill. ; 24 cm.
ISBN: 0471118494 (pbk. : acid-free paper)

CALL NUMBER: QA76.76.H94 G73 1995 FT MEADE
Copy 1
-- **Request in:** Main or Science/Business Reading Rms - STORED OFFSITE
-- **Status:** Not Charged

CALL NUMBER: QA76.76.H94 G73 1995 FT MEADE
Copy 2
-- **Request in:** Main or Science/Business Reading Rms - STORED OFFSITE
-- **Status:** Not Charged

EXHIBIT L-2

```
<!-- Start of HTML header portion -->
<link rel="stylesheet" type="text/css" href="/css/basic.css" />
<link rel="stylesheet" type="text/css" href="/css/timer.css" />
<script type="text/javascript" src="/js/valid.js"></script>
<script type="text/javascript" src="/js/timer.js"></script>
<script type="text/javascript" src="/js/zdragdroplib.js"></script>
<table width="98%" border="0" cellspacing="0" cellpadding="0" align=center>
<tr>
<td colspan=5 align=left><table width="100%" cellpaddiing=0 cellspacing=0 border=0><tr>
<td width="53%"><font face="Arial, Helvetica, sans-serif" size="-1"><a href="http://www.loc.gov/">The
Library of Congress</a></font></td>
<td align=right width="47%"><font face="Arial, Helvetica, sans-serif" size="-1">&gt;&gt;
<a href="http://authorities.loc.gov/"> Go to Library of Congress Authorities</a></font></td>
</tr></table></td>
</tr>
<tr height=55>

<td width=57><img src="/images/banner-rose-left.gif" width="57" height="55" alt=""></td>

<td width="50%" background="/images/banner-back.gif"> </td>

<td width=488><A HREF="/webvoy.htm"><img src="/images/banner-center.gif" width="488" height="55" alt="Start Over" border="0"></a></td>

<td width="50%" background="/images/banner-back.gif"> </td>

<td width=60><img src="/images/banner-rose-right.gif" width="60" height="55" alt=""></td>
</tr>
</table><!-- End of HTML header portion -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
<meta http-equiv="Content-Type" Content="text/html;charset=UTF-8">
<TITLE>Brief Record (Library of Congress Online Catalog)</TITLE>
</HEAD>
<BODY  bgcolor="#ffffff" text="#000000" link="#006666" vlink="#993300"><CENTER>
<FONT face=Default size=3><A HREF="/help/disphlp1.htm"><IMG ALIGN="MIDDLE" BORDER=0 ALT="Help" SRC="/images/UpHelp.gif"></A><A HREF="/cgi-bin/Pwebrecon.cg
><IMG ALIGN="MIDDLE" BORDER=0 ALT="Headings List" SRC="/images/UpHeadings.gif"></A><A HREF="/cgi-bin/Pwebrecon.cgi?ti=1,0&Search%5FArg=graham%2C%20ian%20s
><IMG ALIGN="MIDDLE" BORDER=0 ALT="Titles List" SRC="/images/UpTitles.gif"></A><INPUT TYPE=HIDDEN NAME=BCKT VALUE=><A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=RE
<INPUT TYPE=HIDDEN NAME=Search_Arg VALUE="graham, ian s."><INPUT TYPE=HIDDEN NAME=Search_Code VALUE="NAME_"><INPUT TYPE=HIDDEN NAME=SL VALUE=""><INPUT TYP
<TR>
<TD><font size="-1" color="#006666"><b>DATABASE:</b></font> Library of Congress Online Catalog</TD>
<BR>
</TR>
<TR>
<TD ALIGN=LEFT><font size="-1" color="#336666"><b>YOU SEARCHED</b></font>: Author/Creator Browse = graham, ian s.</TD>
</TR>
<TR>
<TD ALIGN=LEFT><font size="-1" color="#336666"><b>SEARCH RESULTS</b></font>: Displaying 6 of 8.</TD>
</TR>
</TABLE>
<CENTER>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=5&ti=1,5&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Previous Screen or Record" SRC="/images/s-prev.gif"></A>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=7&ti=1,7&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Next Screen or Record" SRC="/images/s-next.gif"></A>
</CENTER>
<CENTER><IMG SRC="/images/view1a.gif" ALIGN=MIDDLE BORDER=0 ALT="Brief Record"><A HREF="/cgi-bin/Pwebrecon.cgi?v2=6&ti=1,6&SEQ=20080228120449&Search%5FArg
><IMG SRC="/images/view2.gif" ALIGN=MIDDLE BORDER=0 ALT="Subjects/Content"></A><A HREF="/cgi-bin/Pwebrecon.cgi?v3=6&ti=1,6&SEQ=20080228120449&Search%5FArg
><IMG SRC="/images/view3.gif" ALIGN=MIDDLE BORDER=0 ALT="Full Record"></A><A HREF="/cgi-bin/Pwebrecon.cgi?v4=6&ti=1,6&SEQ=20080228120449&Search%5FArg=grah
><IMG SRC="/images/view4.gif" ALIGN=MIDDLE BORDER=0 ALT="MARC Tags"></A></CENTER><P>
<CENTER><H3></H3></CENTER>
<CENTER>
<H4>
<I>
<STRONG><BDO dir="LTR">The HTML sourcebook / Ian S. Graham.</BDO></STRONG>
</I>
</H4>
</CENTER>
<INPUT TYPE=HIDDEN NAME=RID VALUE=2071355><TABLE>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>LC Control No.:</TH>
<TD dir="ltr">
    94047369 </TD></TR><TR><TH NOWRAP ALIGN=RIGHT VALIGN=TOP>LCCN Permalink:</TH>
<TD><a href="http://lccn.loc.gov/   94047369 " target="_blank">http://lccn.loc.gov/   94047369 </a></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Type of Material:</TH>
<TD>
Book (Print, Microform, Electronic, etc.)</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Personal Name:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=Author&SEQ=20080228120449&PID=3898&SA=Graham,+Ian+S.,+1955-"> Graham, Ian S., 1955- </A></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Main Title:</TH>
<TD dir="ltr">
 The HTML sourcebook / Ian S. Graham.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Published/Created:</TH>
<TD dir="ltr">
 New York : Wiley, c1995.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>Description:</TH>
<TD dir="ltr">
 xvi, 416 p. : ill. ; 24 cm.</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>ISBN:</TH>
<TD dir="ltr">
 0471118494 (pbk. : acid-free paper)</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
```

```
<TD></TD></TR>
<TR><TD COLSPAN=2><A name="D1"></A><HR WIDTH=150></TD></TR>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>CALL NUMBER:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=CallNumber&SEQ=20080228120449&PID=3898&SA=QA76.76.H94"> QA76.76.H94 G73 1995 FT MEADE</A></TD>
</TR>
<TR>
<TD></TD>
<TD dir="ltr">Copy 1</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Request in:</TH>
<TD>
Main or Science/Business Reading Rms - STORED OFFSITE</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
<TD></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Status:</TH>
<TD>
Not Charged</TD></TR>
<TR><TD COLSPAN=2><A name="D2"></A><HR WIDTH=150></TD></TR>
<TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>CALL NUMBER:</TH>
<TD dir="ltr">
<A HREF="/cgi-bin/Pwebrecon.cgi?SC=CallNumber&SEQ=20080228120449&PID=3898&SA=QA76.76.H94"> QA76.76.H94 G73 1995 FT MEADE</A></TD>
</TR>
<TR>
<TD></TD>
<TD dir="ltr">Copy 2</TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Request in:</TH>
<TD>
Main or Science/Business Reading Rms - STORED OFFSITE</TD></TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP></TH>
<TD></TD>
</TR>
<TR>
<TH NOWRAP ALIGN=RIGHT VALIGN=TOP>-- Status:</TH>
<TD>
Not Charged</TABLE>
<BR><CENTER>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=5&ti=1,5&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Previous Screen or Record" SRC="/images/s-prev.gif"></A>
<A HREF="/cgi-bin/Pwebrecon.cgi?v1=7&ti=1,7&SEQ=20080228120449&Search%5FArg=graham%2C%20ian%20s%2E&Search%5FCode=NAME%5F&CNT=25&PID=3898&SID=1"
><IMG ALIGN=MIDDLE BORDER=0 ALT="Next Screen or Record" SRC="/images/s-next.gif"></A>
</CENTER>
<BR><INPUT TYPE=HIDDEN NAME=BIB VALUE=2071355><INPUT TYPE=HIDDEN NAME=PID VALUE="3898">
<INPUT TYPE=HIDDEN NAME=SEQ VALUE="20080228120449">
<CENTER>
<TABLE BORDER=1 BGCOLOR="#FFFFDD">
  <TR>
      <TH>Save, Print or Email Records (View <a href="/help/savemail.htm">Help</a>)</TH>
   </TR>
  <TR>
     <TD> Select Download Format:
        <SELECT NAME=RD>
            <OPTION VALUE=0>Text (Brief Information)</OPTION>
            <OPTION VALUE=1>Text (Full Information)</OPTION>
            <OPTION VALUE=2>MARC (non-Unicode/MARC-8)</OPTION>
            <OPTION VALUE=3>MARC (Unicode/UTF-8)</OPTION>
        </SELECT>
<INPUT TYPE="SUBMIT" NAME="SAVE" VALUE="Press to SAVE or PRINT" onClick="document.FAKEFORM.PS.value='1'">
</TD>  </TR>
  <TR>
               <TD><font size="-1" color="#336666"><b>Email Text (Full Info) to:</b></font><br>
                       <INPUT NAME="MAILADDY" MAXLENGTH=45 SIZE=32 onKeyPress="return submitenter(this,event)">
<SCRIPT  LANGUAGE="Javascript">
function submitenter(myfield,e){
        var keycode
        if(window.event) keycode = window.event.keyCode
        else if(e) keycode = e.which
   else return true
        if(keycode == 13){
var v1 = document.frm.MAILADDY.value;errorOccured = false;if(v1.length == 0){errorOccured = true; alert("No email address has been entered. Please enter a
document.frm.MAILADDY.focus();return true;}
if(!errorOccured){document.frm.EMAILADDRESS.value = "EMAIL"
document.frm.submit();return false;}}
else return true
}</SCRIPT>
<INPUT TYPE="button" NAME="EMAIL" VALUE="Press to SEND EMAIL" onClick=checkValues()>
<INPUT TYPE="HIDDEN" NAME="EMAILADDRESS" VALUE="None">
  </TR>
</TABLE></CENTER>
<SCRIPT  LANGUAGE="Javascript">
function checkValues(){var v1 = document.frm.MAILADDY.value;errorOccured = false;var email = /(\w+)@(\w+)/;
if(v1.length == 0){errorOccured = true; alert("No email address has been entered. Please enter an email address")
document.frm.MAILADDY.focus();}
else if(email.test(document.frm.MAILADDY.value)== false){
errorOccured = true; alert("Please enter a valid email")
document.frm.MAILADDY.focus();}
if(!errorOccured){document.frm.EMAILADDRESS.value = "EMAIL"
document.frm.submit()}}</SCRIPT>
</FORM><BR><FORM NAME="FAKEFORM">
<INPUT TYPE="HIDDEN" NAME="PS" VALUE='0'>
</FORM>
<CENTER>
<FONT face=Default size=3><A HREF="/help/disphlp1.htm"><font size="-1" face="Arial, Helvetica">Help</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi
><font size="-1" face="Arial, Helvetica">History</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?hd=0,0&PID=3898&HEADBROWSE=0&HIST=0&SEQ=2008022812
><font size="-1" face="Arial, Helvetica">Headings</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?ti=1,0&Search%5FArg=graham%2C%20ian%20s%2E&Search
A><font size="-1" face="Arial, Helvetica">Titles</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=REQUESTBIB&SEQ=20080228120449&PID=3898"
><font size="-1" face="Arial, Helvetica">Request</font></A> - <A HREF="/cgi-bin/Pwebrecon.cgi?PAGE=pbLogon&PID=3898"><font size="-1" face="Arial
><font size="-1" face="Arial, Helvetica">Exit</font></A></FONT></CENTER></BODY>
```

```
<!-- Start of HTML footer portion -->
<hr noshade size=1>
<table width="100%" border="0" cellspacing="6" cellpadding="4">
<tr valign="middle">
<td width="34%"><img src="/images/small-disc.jpg" alt="" width="90" height="87" hspace="2" vspace="0" align="left">
<strong><font face="Arial, Helvetica, sans-serif" size="-1">The Library
of Congress</font></strong><font face="Arial, Helvetica, sans-serif" size="-1"><br>
URL: <a href="http://www.loc.gov/">http://www.loc.gov/</a><br>
<i>Mailing Address: </i></font><font face="Arial, Helvetica, sans-serif" size="-1"><br>
101 Independence Ave, S.E.<br>
Washington, DC 20540</font></td>
<td width="36%" align="center" bgcolor="#FFFFdd"><font size="-1" face="Arial, Helvetica, sans-serif"><b>Catalog/authority
record errors?<br>
</b><a href="http://www.loc.gov/help/contact-libarch-report.html">Use our
Error Report Form</a><br>
<strong>Questions about searching?</strong><br>
<a href="http://www.loc.gov/rr/askalib/ask-digital.html">Ask a Librarian</a></font></td>
<td width="30%" align="right">
<p><font face="Arial, Helvetica, sans-serif" size="-1"><b>Library of Congress
Online Catalog</b><br>
<i>URL:</i> <a href="http://catalog.loc.gov/">http://catalog.loc.gov/</a><br>
<b>Library of Congress Authorities</b><br>
<i>URL:</i> <a href="http://authorities.loc.gov/">http://authorities.loc.gov/</a></font></p>
</td>
</tr>
</table>
<script type="text/javascript">
<!--//
document.write(getWarning());
//-->
</script>
<div id="warningDiv"></div>
<img src="/images/load.gif" onLoad="initTimer(); getElements();" />
<!-- End of HTML footer portion -->
</HTML>
```

EXHIBIT M

US005377323A

United States Patent [19]

Vasudevan

[11] Patent Number: 5,377,323

[45] Date of Patent: Dec. 27, 1994

[54] **APPARATUS AND METHOD FOR A FEDERATED NAMING SYSTEM WHICH CAN RESOLVE A COMPOSITE NAME COMPOSED OF NAMES FROM ANY NUMBER OF DISPARATE NAMING SYSTEMS**

[75] Inventor: **Rangaswamy Vasudevan**, Sunnyvale, Calif.

[73] Assignee: **Sun Microsytems, Inc.**, Mountain View, Calif.

[21] Appl. No.: **760,044**

[22] Filed: **Sep. 13, 1991**

[51] **Int. Cl.**[5] **G06F 13/00**

[52] **U.S. Cl.** **395/200**; 395/600; 395/325; 364/284.3; 364/284.4; 364/DIG. 1

[58] **Field of Search** 395/600, 200, 325; 380/23; 364/419.08

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,435,769	3/1984	Nagano et al.	364/464
4,912,637	3/1990	Sheedy et al.	364/300
5,136,716	8/1992	Harvey et al.	395/800
5,187,787	2/1993	Skeen et al.	395/600
5,224,163	6/1993	Gasser et al.	380/30
5,239,577	8/1993	Bates et al.	379/201

OTHER PUBLICATIONS

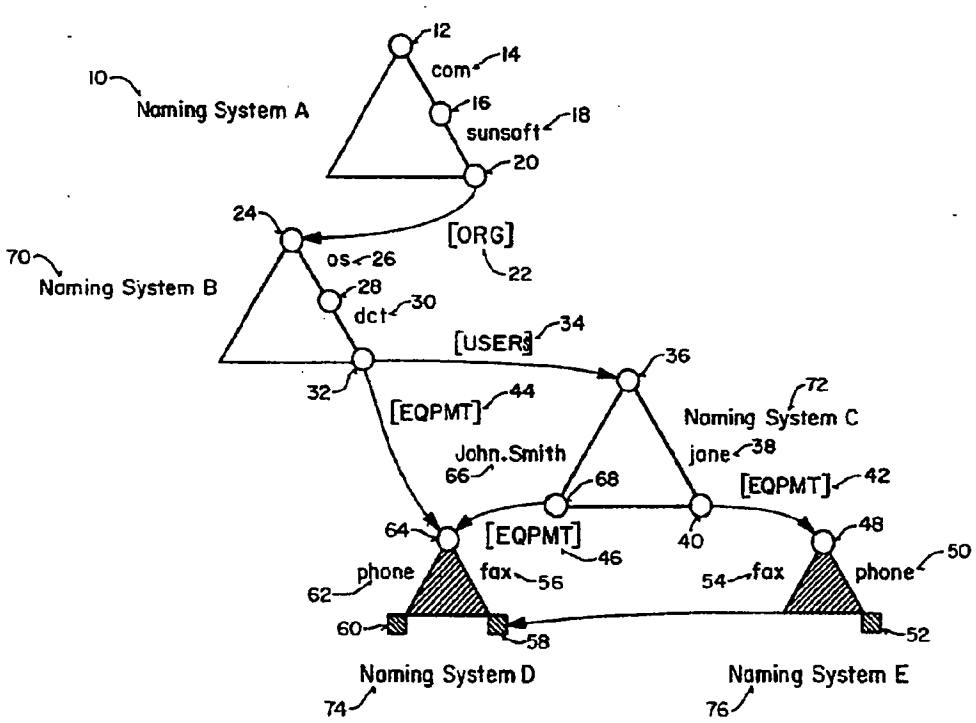
Honeyman et al., "Parsing Ambiguous Addresses for Electronic Services", *Software–Practice and Experience*, vol. 17, No. 1, pp. 51–60 (Jan., 1987).

Primary Examiner—Thomas C. Lee
Assistant Examiner—Paul Lintz
Attorney, Agent, or Firm—Erwin J. Basinski

[57] **ABSTRACT**

In a distributed computing environment, an apparatus and method for a federated Naming System which can resolve Composite Names comprised of Names from an arbitrary number of disparate Naming Systems. A syntax for Composite Names is defined as well as necessary operations to directly resolve such Composite Names without the need for customized agents or gateways.

9 Claims, 6 Drawing Sheets



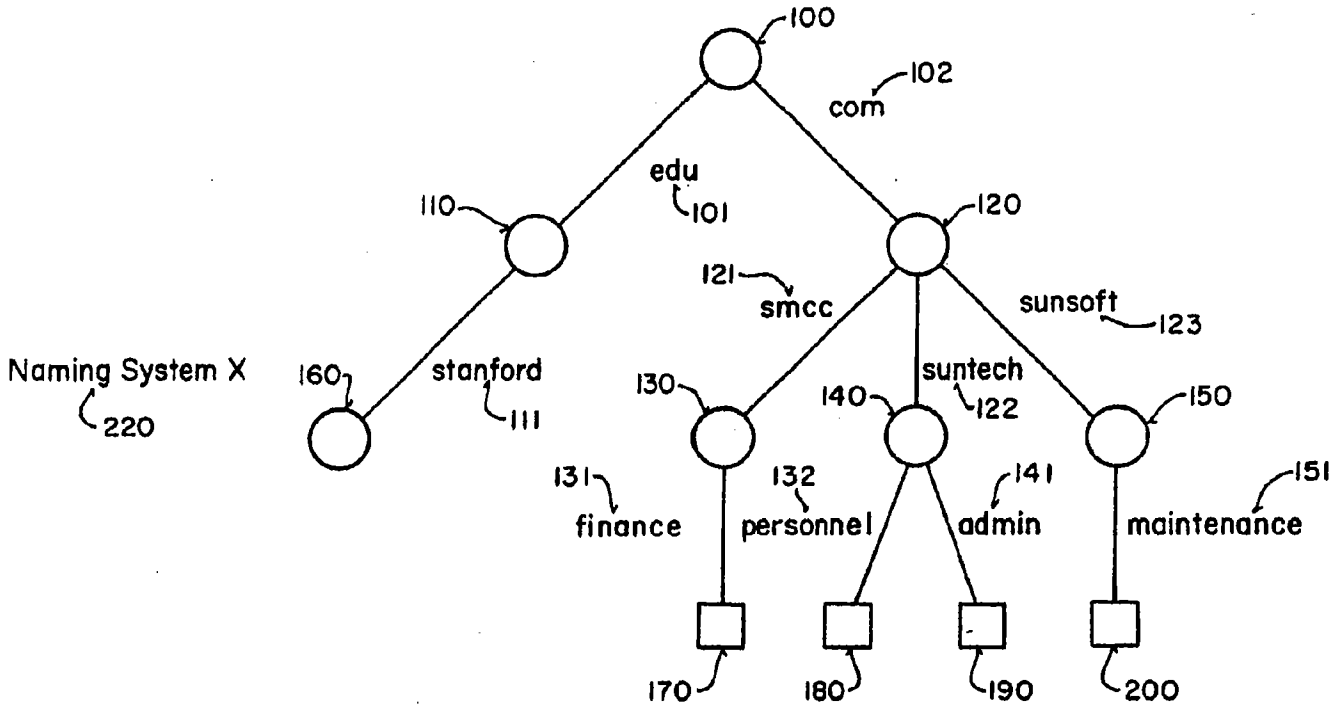

100
102
com
edu
101
110
120
121
smcc
sunsoft
123
Naming System X
220
160
stanford
111
130
140
suntech
122
150
131
finance
132
personnel
141
admin
151
maintenance
170
180
190
200

FIGURE 1

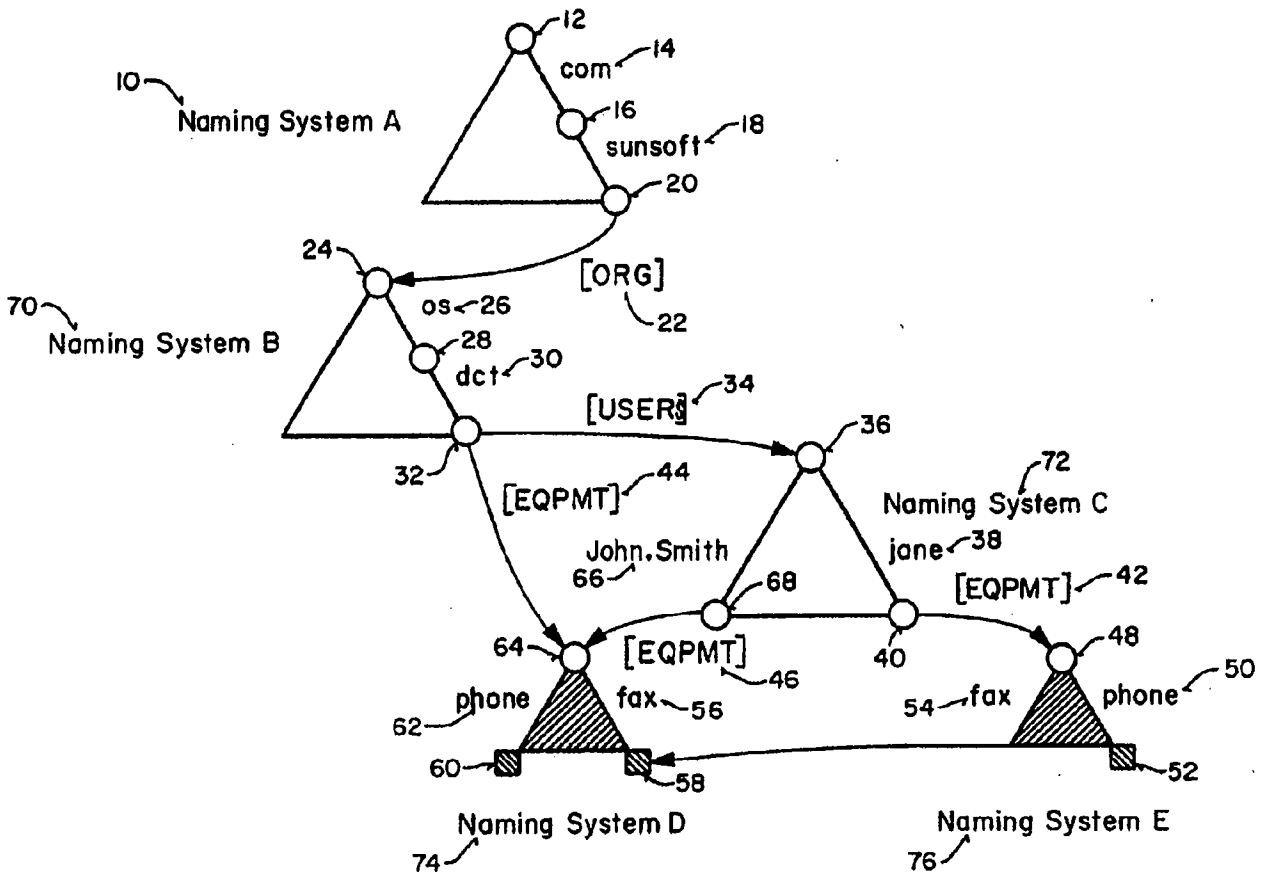

12
com
14
10
Naming System A
16
sunsoft
18
20
24
[ORG]
22
70
os
26
Naming System B
28
dct
30
[USERS]
34
36
32
[EQPMT]
44
72
Naming System C
John.Smith
66
jane
38
68
[EQPMT]
42
64
[EQPMT]
46
40
48
50
phone
62
fax
56
54
fax
phone
60
58
52
Naming System D
74
Naming System E
76

FIGURE 2

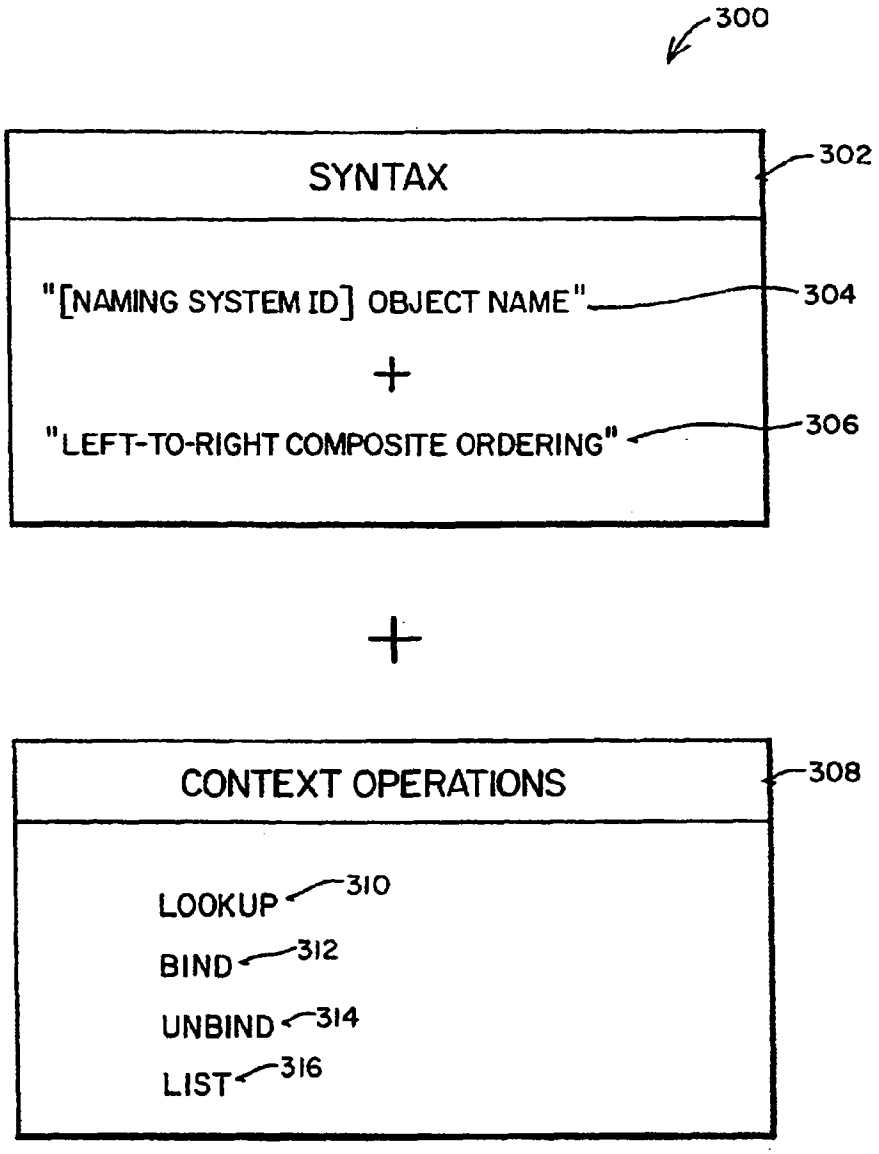

300
SYNTAX
302
"[NAMING SYSTEM ID] OBJECT NAME"
304
+
"LEFT-TO-RIGHT COMPOSITE ORDERING"
306
+
CONTEXT OPERATIONS
308
LOOKUP
310
BIND
312
UNBIND
314
LIST
316

FIGURE 3

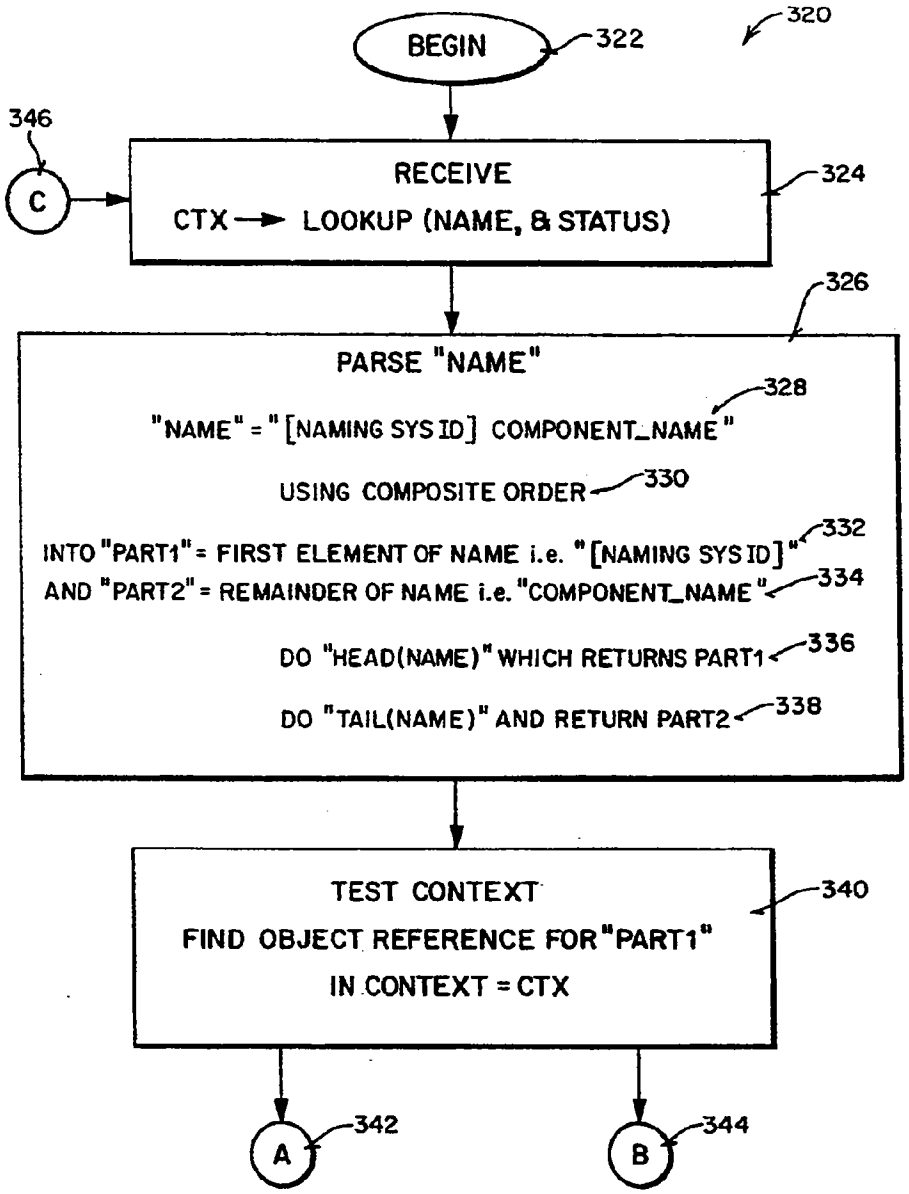

320
BEGIN
322
346
C
RECEIVE
CTX → LOOKUP (NAME, & STATUS)
324
326
PARSE "NAME"
328
"NAME" = "[NAMING SYS ID] COMPONENT_NAME"
USING COMPOSITE ORDER
330
INTO "PART1" = FIRST ELEMENT OF NAME i.e. "[NAMING SYS ID]"
332
AND "PART2" = REMAINDER OF NAME i.e. "COMPONENT_NAME"
334
DO "HEAD(NAME)" WHICH RETURNS PART1
336
DO "TAIL(NAME)" AND RETURN PART2
338
TEST CONTEXT
FIND OBJECT REFERENCE FOR "PART1"
IN CONTEXT = CTX
340
A
342
B
344

FIGURE 4

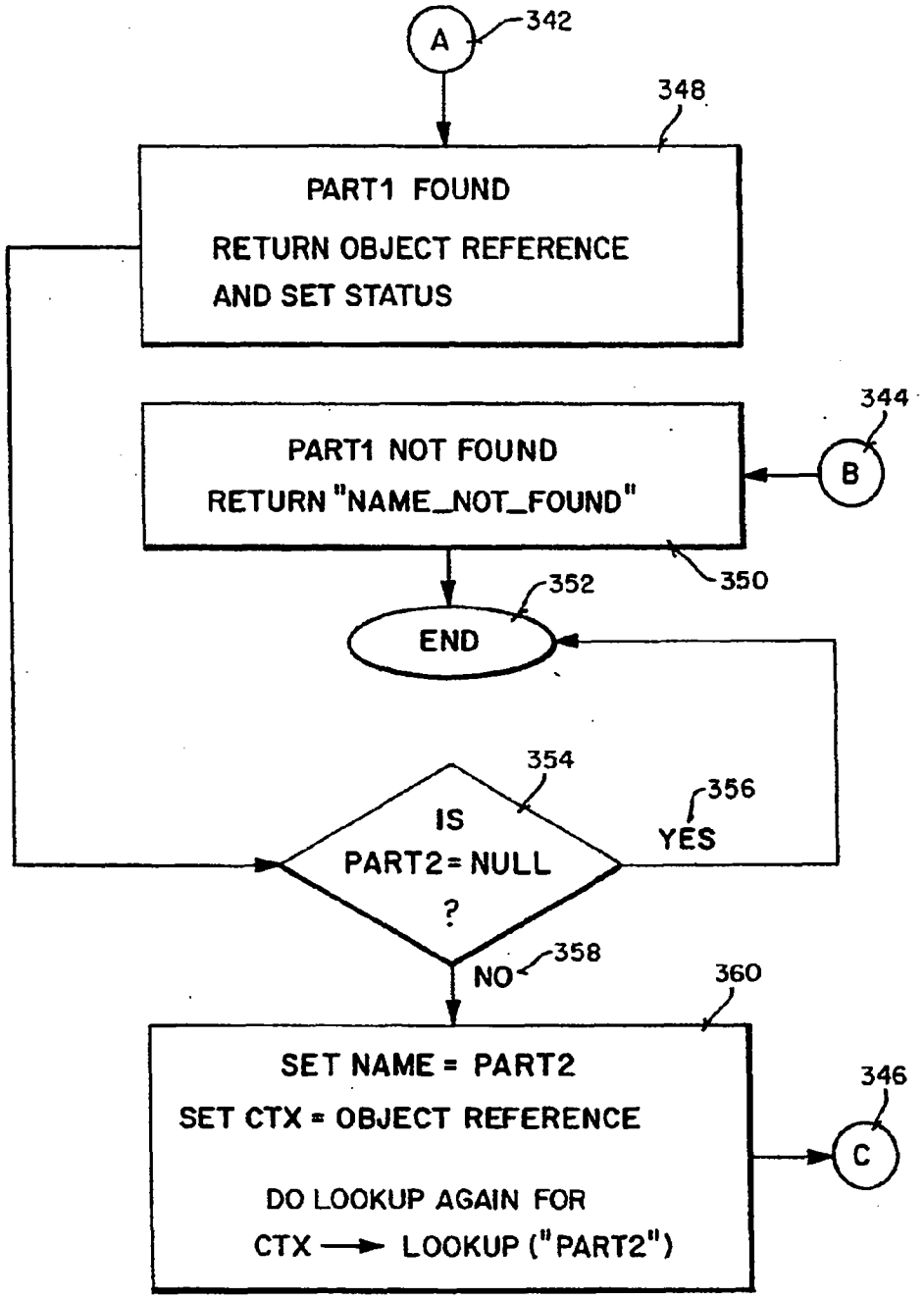

A
342
348
PART1 FOUND
RETURN OBJECT REFERENCE
AND SET STATUS
344
PART1 NOT FOUND
RETURN "NAME_NOT_FOUND"
B
350
352
END
354
IS
PART2 = NULL
?
356
YES
NO
358
360
SET NAME = PART2
SET CTX = OBJECT REFERENCE
DO LOOKUP AGAIN FOR
CTX → LOOKUP ("PART2")
346
C

FIGURE 5

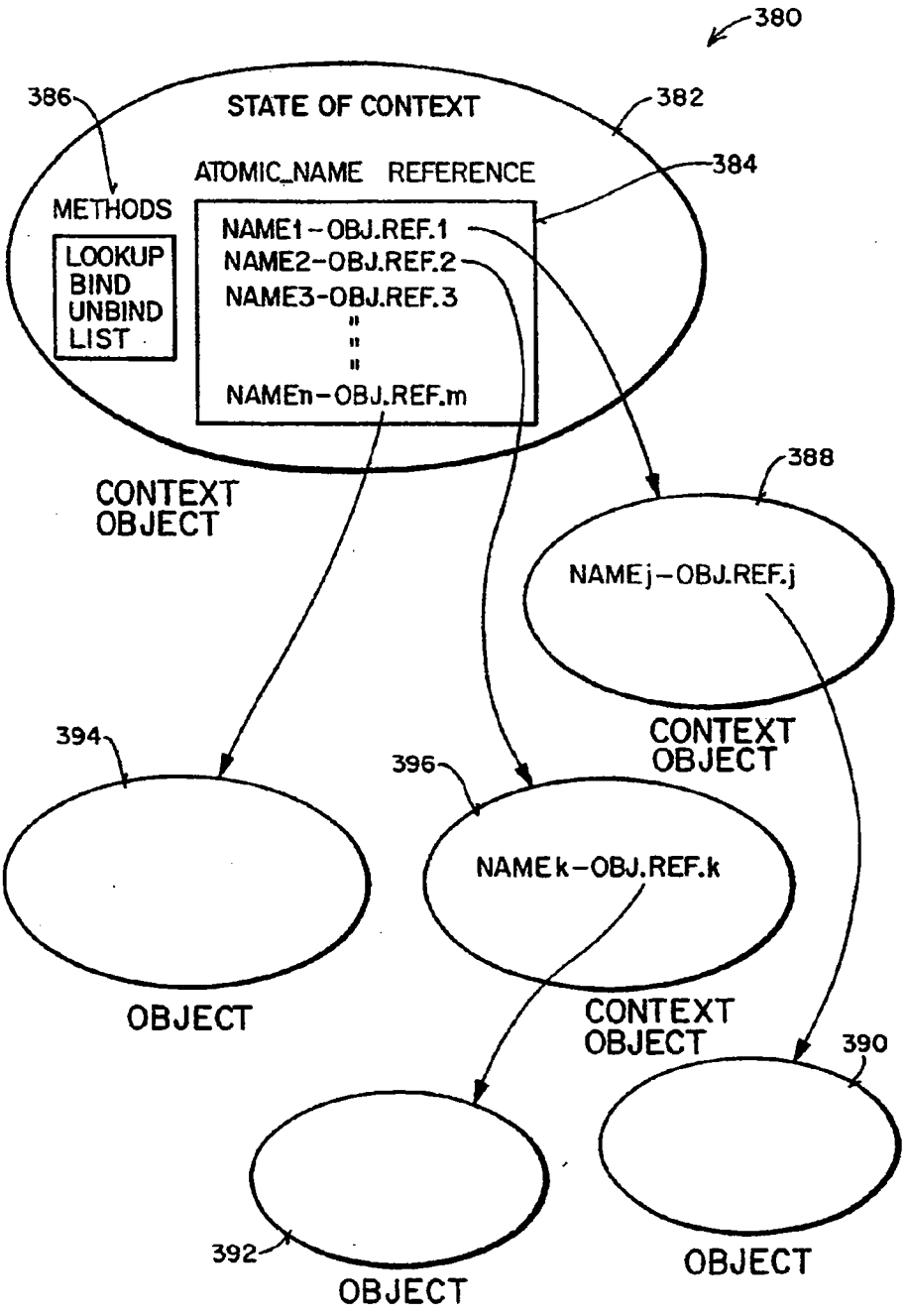

380
386
STATE OF CONTEXT
382
ATOMIC_NAME REFERENCE
384
METHODS
LOOKUP
BIND
UNBIND
LIST
NAME1-OBJ.REF.1
NAME2-OBJ.REF.2
NAME3-OBJ.REF.3
NAMEn-OBJ.REF.m
CONTEXT OBJECT
388
NAMEj-OBJ.REF.j
CONTEXT OBJECT
394
396
NAMEk-OBJ.REF.k
OBJECT
CONTEXT OBJECT
390
392
OBJECT
OBJECT

FIGURE 6

APPARATUS AND METHOD FOR A FEDERATED NAMING SYSTEM WHICH CAN RESOLVE A COMPOSITE NAME COMPOSED OF NAMES FROM ANY NUMBER OF DISPARATE NAMING SYSTEMS

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates to Object-Based Distributed Computing Systems, and more particularly to the field of naming systems.

2. Description of Related Art and Background

A fundamental facility in any computing system is the Naming Service (the "Naming Service"). A Naming Service is the means by which names are associated with objects, and by which objects are found given only their names. A Name ("Name") is a sequence of one or more atomic names, where an atomic name ("Atomic Name") is a name defined by a naming convention which, for example, may be a sequence of one or more characters. For example, "user/John. Smith/foo" is a Name and "user", "John. Smith", and "foo" are Atomic Names. An object ("Object") is generally a combination of data and operators or operational routines related to that data. Naming Services usually provide operations for

associating, or binding, names to objects
finding the objects bound to given names
removing name association or bindings
querying, renaming, etc.

In early versions of computing systems, the naming schemes used were not the result of well defined naming services but rather an incidental set of rules for naming things created by the system designer. Over the past decade more systematic and formal approaches to the development of computer systems designs have led to the creation of discrete or standalone computer services such as file systems, directory services, databases, etc. Each of these services, in many cases, have created their own set of rules for naming Objects within the service, and for use by clients.

Most of the early Naming Services did not communicate with each other. Clients had to use the services separately. In those rare cases where such services had to be used together, communication was by means of a "Composite Name". A "Composite Name" is a sequence of names from more than one Naming System. In today's UNIX® systems (UNIX is a registered trademark of UNIX Systems Laboratories, Inc., a wholly-owned subsidiary of Novell, Inc., certain applications, such as "mount" and "rcp", implement resolution of Composite Names with components from different naming systems. To do this, these applications may use a special delimiter character to separate Names in the Composite Name, enabling them to resolve the Composite Name to the corresponding Object by using each component naming system. Unfortunately, the use of such Composite Names is limited to the small set of applications that are able to resolve them. Also, such applications are typically restricted to only a small and fixed number of Naming Systems. Adding a new type of Naming System would require all of the applications to be changed.

A serious problem with the use of Composite Names today is the lack of uniformity and transparency. The user must be aware of which commands accept Composite Names, and their required syntax. For example, the composite name "sylvan:/temp/foo" is acceptable to the UNIX command "rcp", but it is not accepted by the command "cp".

More recently, the increased use of computers in business and at home, due to reduced costs and increased computer literacy, has led to increased demand for access between computer systems. Portable computers allow users to move them freely about the world but these users continue to require constant access to their host applications, files, databases and electronic mail via connections to other computer systems. New networks of computers require more access to more and more disparate networks and related systems. These increasing demands have produced the present focus on Distributed Computer Systems and on methods to interoperate these systems. This focus on interoperability is on developing means for easy access from one computing system to another, regardless of whether the two systems have different naming systems, different operating systems, different file systems, different databases, etc., with minimal cost of modifying any system to interoperate with another.

Moreover, widespread deployment of fiberoptic communication lines, together with high speed packet-switching technologies, are greatly increasing the capacity and performance of wide-area networks. These advances, coupled with new object-based technologies, create a tremendous need for systems and methods which will allow systems to communicate with each other efficiently.

A major stumbling block in developing such easy access between systems is the disparate naming conventions and naming operations used by the various individual file systems, databases, operating systems, communications systems, etc.

Several approaches have been and are being developed to address this problem of dealing with disparate naming systems. An approach for handling Composite Names is provided by the Open Software Foundation TM /Distributed Computing Environment (the "OSF/DCE"). Open Software Foundation ("OSF") is a trademark of Open Software Foundation, Inc.

The OSF/DCE approach to this problem of handling the existing disparate naming systems is through the use of Composite Names.

OSF/DCE provides a method of handling composite names composed of three levels of naming systems: 1) the global directory service, either CCITT's X.500 or Internet's Domain Naming Service ("DNS") that names cells; 2) the OSF/DCE Cell Directory Service "CDS") that names users, file servers, databases, and other servers in a cell; and 3) the naming system of OSF's Distributed File System ("DFS") and other services named by CDS.

For example, the following is a Composite Name handled by OSF/DCE,

"/. . . / C=US/O=OSF/OU=Cambridge/fs/user/-John.Smith/foo"

where

/=name of the local computer host's root

. . . =name of the global root

C=US/O=OSF/OU=Cambridge=X.500 name of a cell root

fs=CDS name of the DFS root

user/John.Smith/foo=DFS UNIX file name.

This approach of OSF/DCE permits Composite Names spanning two or three Naming Systems (three in the above example), and provides a significant advance in handling disparate Naming Systems. However, this approach uses agents that are customized to specific naming systems. This method lacks the flexibility and scalability that will be required by Object-Based distributed systems in the future. A change to one of OSF/-DCE's Naming Systems, or the addition of a new type of Naming System requires changes to existing agents or the addition of a new customized agent. Also, the future need will be to support Composite Names that span an arbitrary number of Naming Systems, not just a fixed number such as two or three Naming Systems.

At present, the development of Object-Based Distributed Systems has just begun and the expected proliferation of such systems and the corresponding increase in the number and type of "Objects" requiring "Names" and Name resolution makes the development of efficient and flexible Naming Systems of paramount importance.

Accordingly, the present invention defines a model which can describe any Naming System and its related characteristics, defines a method of federating an arbitrary number of disparate Naming Systems with minimal computing cost and effort to participate in the federation, and defines an apparatus and method for using this federated Naming System to perform the functions of name resolution in an Object-Based Distributed Computing Environment. It is noted that existing Naming Systems which are not Object-Based may be able to participate in a federated system such as is disclosed herein, but only at some expense through the use of customized gateways.

The prior art does not define Naming Systems for Composite Name resolution in a federation of an arbitrary number of disparate Naming Systems.

SUMMARY OF THE INVENTION

Goals of the present invention are to:

Provide Clients with a single uniform Composite Name resolution interface to a Federated Naming System;

Allow an autonomous Naming System to participate in the federation with minimal cost and effort and with no impact on existing Naming Systems;

Provide a system which allows the addition of new services with their own Naming Systems without requiring changes to Clients or to other existing Naming Systems in the Federation;

Provide a system which does not require intermediate customized Gateways or customized Agents to assist in the resolution of Composite Names, thereby resulting in better performance and fewer fault tolerance issues.

To accomplish these goals, the invention provides, in a distributed computing environment, an apparatus and method for a Federated Naming System which can resolve Composite Names comprised of Names from an arbitrary number of disparate Naming Systems, In order to join in a federation of Naming Systems, the individual Naming Systems are required to have within their systems the means for separating a Name into a "head" Name component and a "tail" Name component as defined herein. The method for doing this resolution internally is left to the individual Naming System. Given however, that the individual Naming System has the required parsing methods, they may then join in the specified federation by implementing a federation syntax for Composite Names, and by implementing a Composite Name Lookup operation at a minimum. The invention specifies a method and apparatus for implementing such a federation of Naming Systems and for resolving Federation Composite Names.

With this apparatus and method, systems with disparate Naming Systems may easily federate to provide their individual clients with efficient resolution of Composite Names of Objects located in other systems.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is depiction of a Naming System in the prior art.

FIG. **2** is a representation of a Federated Naming System as described in this disclosure.

FIG. **3** illustrates the requirements of the Federated Naming System.

FIGS. **4** and **5** illustrate a block diagram of the Lookup operation.

FIG. **6** illustrates the context object and relationships with other objects.

DETAILED DESCRIPTION OF THE INVENTION

An Apparatus and Method are disclosed which provides an efficient name resolution system for a federation of any number of disparate Naming Systems in an Object-Based Distributed Computing Environment. A model of a Naming System is described which can define the characteristics of any Naming System. Within the context of this Naming System model, a method for federating any number of disparate Naming Systems is described, including an apparatus and method for name resolution of a Composite Name composed of any number of such discrete Naming Systems.

NAMING MODEL

The following object-based model defines Naming Systems in abstract terms.

A "Name" is a sequence of one or more atomic names.

An "Atomic Name" is a value specified by a naming convention, which, for example, could be a sequence of one or more characters. Every Name is defined by a "Naming Convention" which is a set of syntactic rules which govern the form of a Name. A naming convention defines all possible Atomic Names and enables the definition of operations for parsing any Name to produce its sequence of Atomic Names. The naming convention also enables the definition of a relation for determining if two Names are equivalent.

The two basic name parsing operations are, "head(-Name)" that returns the first atomic component of "Name", and "tail(Name)" that returns the remainder of "Name" after "head(Name)" is removed. For example, in parsing the name "user/John.Smith/foo" the operation "head (user/John. Smith/foo)" would return "user" (assuming "user" is defined as the "head" by the particular naming convention); and the operation "tail(user/John.Smith/foo)" would return "John.Smith/foo" (See Block 326 of FIG. 4, for example.

A "Name Space" is the set of all possible Names generated according to a naming convention.

A "Binding" is an association of an Atomic Name with an Object reference.

An "Object Reference" is an address or device used to invoke operations on an Object.

A "Context" is an object whose state is a set of bindings with distinct Atomic Names. Every Context type has an associated Naming convention. A Binding can associate any type of object to an Atomic Name. Thus a Context can contain bindings of Atomic Names to other Contexts. In this way, file systems and directory services are hierarchically structured (see FIG. 6, for example.)

The naming operations provided by a Context include "Lookup", "Bind", "UnBind", "List", and "CreateContext". These are defined as follows for a Context denoted "c", a Name denoted "n", an Atomic Name denoted "an", and an Object reference denoted "s". An invocation of "c.LookUp(n)" is evaluated as follows: Let "s" be the Object reference bound to "head(n) in "c". If "n" is an Atomic Name, then return "s". Otherwise, return "s.LookUp(tail(n))". For example, referring now to FIG. 1, we shall describe how this operation "c.LookUp(n)" would work in an example using an existing Naming System. FIG. 1 depicts a Naming System X of a type that can be found in the prior art. Naming System X 220 contains Context Objects 100, 110, 120, 130, 140, 150, and 160, and Objects 170, 180, 190, and 200. Context 100 contains two bindings, "corn" 102 to Context Object 120, and "edu" 101 to Context Object 110. Context 120 contains three bindings, "smcc" 121 to Context Object 130, "suntech" 122 to Context Object 140, and "sunsoft" 123 to Context Object 150. Context 150 contains only one binding, "maintenance" 151 to Object 200. To resolve a Name "maintence.sunsoft.com" from Naming System X 220 in Context 100, we invoke on Context 100 the LookUp operation as follows

context 100->LookUp ("maintenance.sunsoft.com")

In Context 100 the following operations are performed:

head("maintenance.sunsoft.com") which returns "com"; and

tail("maintenance.sunsoft.com") which returns "maintenance.sunsoft".

Since the Context 100 knows that "com" (which was returned by the "head" operation), is bound to a reference to Context 120, and since the "tail" operation returned a non-null value (i.e. "maintenance.sunsoft"), the resolution process continues by invoking "LookUp" again but this time relative to Context 120 and with the reduced Name "maintenance.sunsoft" that was returned by "tail". So, in Context 120 the following operations are performed:

head("maintenance.sunsoft") which returns "sunsoft"; and

tail("maintenance.sunsoft") which returns "maintenance".

Since the Context 120 knows that "sunsoft" (which was returned by the "head" operation), is bound to a reference to Context 150, and since the "tail" operation returned a non-null value (i.e. "maintenance"), the resolution process continues by invoking "LookUp" again but this time relative to Context 150 and with the reduced Name returned by "tail". So, in Context 150 the following operations are performed:

head("maintenance") returns "maintenance"; and

tail("maintenance") returns a null value.

Since the "tail" operation returned a null value the LookUp operation terminates, returning eventually to the Client, the reference to Object 200, which Context 150 knows is bound to "maintenance". (See FIG. 6, for example.)

An invocation of "c.Bind(an,s)" adds the new binding (an,s)to "c". An invocation of "c.UnBind(an)" removes "an's" binding from "c". An invocation of "c.List()" returns the set of atomic names bound in "c". (See FIG. 6 Block 386.)

An invocation of "c.CreateContext(an)" creates a new context object "c'", and adds the new binding (an,c') to "c".

A "Naming System" is a set of context objects that can be represented by a directed graph, where each node in the graph is a context, and an arc from one node to another indicates the existence of a binding. For example, an arc from node a to node b indicates the existence of a binding (an,b) in a. A context is called a "Root Context" if every other Context in the Naming System can be named relative to the Root Context. Typically, a Naming System has a single Root Context.

It should be obvious from the above model of a Naming System, the model can be used to describe any existing Naming System. It should also be noted, however, that existing systems have their own methods for parsing names in their systems but they may be considered equivalent to the "Lookup" operations described in the present invention. The model shall now be extended to include multiple disparate Naming Systems in order to describe in detail the present invention and the presently preferred embodiment thereof.

FEDERATED NAMING SYSTEMS

A "Federated Naming System" is an aggregation of autonomous Naming Systems that cooperate through a standard interface and protocol to implement name resolution for Composite Names. The contexts in a federated Naming System can directly resolve Composite Names without intermediate customized clerks, customized agents, or customized gateways. Each member of a federated Naming System (i.e. the individual Naming Systems) has autonomy in its choice of naming conventions, administrative interfaces (such as security regulations, etc) and its particular set of operations for Name resolution.

Referring now to FIG. 2, a Federated Naming System will be described which is representative of the type of Federated Naming System defined by the present invention. FIG. 2 shows a set of autonomous Naming Systems A 10, B 70, C 72, D 74, and E 76. Naming System A 10 consists of Contexts 12, 16, and 20; Naming System B 70 consists of Contexts 24, 28, and 32; Naming System C 72 consists of Contexts 36, 40, and 68; Naming System D 74 consists of Context 64; and Naming System E 76 consists of Context 48.

The following table shows the Name Bindings in each of the Contexts in FIG. 2.

		Bindings	
Naming System	Context	Atomic Name	Object
A 10	12	com 14	16
A 10	16	sunsoft 18	20
A 10	20	[ORG] 22	24
B 70	24	os 26	28
B 70	28	dct 30	32
B 70	32	[USERS] 34	36

-continued

Naming System	Context	Bindings: Atomic Name	Bindings: Object
B 70	32	[EQPMT] 44	64
C 72	36	John.Smith 66	68
C 72	36	jane 38	40
C 72	68	[EQPMT] 46	64
C 72	40	[EQPMT] 42	48
D 74	64	phone 62	60
D 74	64	fax 56	58
E 76	48	fax 54	58
E 76	48	phone 50	52

The Bindings determine the state of each Naming System **10**, **70**, **72**, **74**, and **76**. The Naming Systems are autonomous in that the naming conventions chosen for each Naming System do not depend on any other. In the example of FIG. **2**, Naming System A **10** uses Atomic Names ordered right-to-left with corresponding Contexts ordered from bottom-to-top. Naming System A **10** also uses the "." character as a delimiter. Thus, the name of Context Object **24** is "[ORG].sunsoft.com". Similarly, in Naming System B **70**, the Name of Context **36** is "os:dct:[USERS]". Naming System B **70** has Naming conventions in which Atomic Names (i.e. "os" **26** and "dct" **30**) are ordered left-to-right corresponding to Contexts from top-to-bottom, and uses the ":" character as a delimiter. In Naming System C **72**, the Name of Context **64** is "John.Smith[EQPMT]". Note that "John.Smith" **66** is an Atomic Name. In Naming System D **64**, Object **58** has the Name "fax" **56**. Naming System D **74** has a flat Naming System in that all names are Atomic Names. Finally, also note that Object **58** also has a name "fax" **54** in Naming System E **76**.

THE PREFERRED EMBODIMENT

In the preferred embodiment at this time, a federation of disparate Naming Systems is designated "The OpenFederation of Naming Systems" (hereinafter "OpenFederation").

In order for a naming system to be a member of the OpenFederation of Naming Systems, it must conform to standards on two fronts.

The Composite Name Syntax

The naming interface to the context object.

Composite Name Syntax

The OpenFederation prescribes a composition syntax for forming Composite Names from component Names from different Naming Systems. In general, the OpenFederation does not specify or restrict the syntax of the component Names from the individual Naming Systems. If there is a conflict between the composition syntax and the naming convention of an individual Naming System, the conflict is resolved by "escaping" such Names.

In the preferred embodiment, the syntax of Composite Names is specified using a pseudo-Backus Naur Form (BNF) as follows:

composite_name: {component}*

component: namingsystem_id_part component_name

namingsystem_id_part: '['namingsystem_id ']'

namingsystem_id: {any char except ']'}+

The Composite Name is specified by ordering the components in left-to-right order. Resolution in the OpenFederation proceeds in this order. Naming System identifiers (namingsystem_ids) separate components of a Composite Name belonging to different Naming Systems. (This is shown in FIG. 3.)

The syntax of "component_name" may vary among Naming Systems. For example, a component Name from an DNS Naming System is defined by the DNS naming syntax.

An example of a Composite Name is

"[dns]eng.sun.com[user]jsmith"

which consists of two components: "[dns]eng.sun.com" and "[user]jsmith".

The portions "[dns]" and "[user]" are Naming System identifiers. "eng.sun.com" and "jsmith" are component Names in the respective Naming Systems.

The Naming System identifiers in Composite Names serve several purposes: (1) as separators between successive components, and (2) in some cases as indicators of the syntactic rules for parsing each component.

Separators between successive components are needed in order to be able to unambiguously separate two components in Naming Systems with conflicting Name syntaxes. The ability to separate such components is necessary to administer the OpenFederation Composite Name resolution scheme at the federation-level given that we wish to handle arbitrary component Naming Systems.

In addition, it is often useful to be able to parse a Name into components without requiring Name resolution. A Composite Name syntax with such separators allows this. To allow independence from a specific Name representation, in the presently preferred embodiment, OpenFederation prescribes a standard set of operations to be used for breaking Composite Names into components. These operations are "Equivalent", "Coun_Components", "Concatenate", "Split", "Get_Suffix", "Copy", and "Free", all of which are described below.

Although a Client can parse a Composite Name into its component Names, in general, clients may not be able to parse the component Names into Atomic Names. For example, given a name like "[org]uw[dt]a|b.c" taken relative to a context "ctx", a client may not be able to break up the component "a|b.c" without invoking a parsing operation provided by the context object obtained by "ctx.LookUp("[org]uw[dt]")".

However, certain Naming System identifiers will be indicators of the syntax of the component Naming System. In the preferred embodiment, OpenFederation will define a set of well-known Naming System ids that are reserved for certain common Naming Systems, like DNS, X.500, and Unix pathnames. Clients will have access to certain standard parsing methods for these well-known Naming Systems. Thus clients will be able to parse those components of a Name that bear well-known Naming System ids.

OpenFederation Operations on Composite Names

In the preferred embodiment,a number of standard operations on Composite Names are implemented for use by clients and by OpenFederation Context Objects for composing, decomposing, and comparing Names. These operations are defined as follows. However, in order to aid in the description of the operations, the following terms are used to refer to parts of Composite Names:

A "prefix" of a Composite Name is any ordered subsequence of consecutive components from the beginning of a Composite Name. A "suffix" of a Composite Name is any ordered subsequence of consecutive components from the terminal end of a composite Name.

Equivalent

In the preferred embodiment, an operation "Equivalent" is defined as follows:

boolean equivalent(IN composite_name_t cname1, IN compo_name_t cname2); Usage: if(equivalent(cname1, cname2)) . . .

If this predicate returns "true" then, the two names "cname1" and "cname2" are syntactically equivalent. The converse is not guaranteed—if the predicate returns "false", the two names may still be equivalent.

Equivalence of two names is sufficient to guarantee that, relative to any one context, the two names cannot refer to different objects.

In general, without resorting to resolution, only exact equality will assure equivalence - however it may be possible in the case of well-known Naming Systems, to determine equivalence accurately without resolution.

This function is useful whenever the client wants to determine whether a given composite name is syntactically the same as another composite name about which it already has information. For example, for caching which helps improve efficiency, "Equivalent" could be used to check whether a given name is already in the cache.

Other operations implemented for the preferred embodiment are as follows:

Count_Components

integer count_components(IN composite_name_t cname); Usage: num_comp=count_components(cname); Returns the number of components in a given composite name, "cname".

Concatenate

composite_name_t concatenate(IN composite_name_t head, IN composite_name_t tail); Usage: head_plus_tail=concatenate(head, tail); Returns a new composite name consisting of the ordered list of components from "head", followed by those in "tail".

Split

split(IN composite_name_t cname, IN integer i, OUT composite_name_t head, OUT composite_name_t tail); Usage: split(three_part_name, 2, &head, &tail); Returns the ordered list of "i" components from "cname" in "head", and the rest ordered in "tail".

This procedure is useful when you know the prefix of a name and want to construct a new name using either the prefix or suffix of the same name. It is probably most useful for extracting the first component or the last component of a composite name.

Get_Suffix

composite_name_t get_suffix(IN composite_name_t cname, IN composite_name_t pname) Usage: whats_left=get_suffix(whole_name, prefix_of) Returns the ordered subsequence of components from "cname" that do not appear in "pname" if "pname" is prefix of "cname". Operation fails if "pname" is not a prefix of "cname".

This procedure is useful in caching, for example, where we may want to determine whether there is cached info,marion about prefixes of a composite name, or the whole composite name.

Copy

composite_name_t copy(IN composite_name_t cname); Usage: new_name=copy(cname); Returns a copy of the given composite name "cname".

Free

free(IN composite_name_t cname); Usage: free(cname); Release the storage used to hold "cname".

A Naming System that wishes to join the OpenFederation federation must adhere to the Composition syntax described above. The Naming System must also provide operations similar to those defined above for "Equivalent", "Count_Components", "Concatenate", "Split", "Get_Suffix", "Copy", and "Free". (See FIG. 3.)

Using this Composition Syntax and the related operations, the standard Naming interface for Contexts in the preferred embodiment OpenFederation is now described.

The OpenFederation Naming Interface

In the preferred embodiment, OpenFederation defines a context interface that includes the "OpenLookUp", "OpenBind", "OpenUnbind", and "OpenList" operations. The interface syntax and semantics are described as follows.

Interface Definition

The "interface definition" in the present embodiment, in terms of pseudocode, is defined as follows:

```
typedef string<unbounded> Composite Name;
const Name NULLNAME=<0>;
typedef sequence <Composite Name,UNBOUNDED> NameList;
const int BIND_SUPERSEDE=0x1
interface Context {
typedef enum (
OK, NOT_CONTEXT, NOT_FOUND, NO_PERMISSION,
ALREADY_BOUND, ILLEGAL_NAME, UNSUPPORTED_OP
) statuscode;
typedef struct {
    statuscode err;
    Composite Name where, rest;
    objref_t ref;
    boolean precisely;
} status;
objref_t OpenLookup( in Composite Name name, out status* );
void OpenBind( in Composite Name name, in objref_t
objref, in int flags, out status* );
void OpenUnbind( in Composite Name name, out status* );
NameList OpenList( in Composite Name name, out status* );
};
```

Requirements

In the preferred embodiment, every OpenFederation Context must provide full support for the "OpenLookup" operation that we describe below. It need not support the "OpenBind", "OpenUnbind", and "OpenList" operations to the extent described. In the case that a terminal context does not support the desired operation, it must return the status code "UNSUPPORTED_OP" and the remaining status information, as described below.

Names

In the preferred embodiment, all of the operations supported by OpenFederation Context objects take Composite Names as arguments. A composite name is interpreted relative to the context object on which the operation is invoked.

The use of "NULLNAME" as a name has a special interpretation. When supplied to a context object, it is a name for that context itself.

In all of the following, "ctx" is an OpenFederation context object. In the preferred embodiment, the following are the required operations.

Lookup

Usage: objecet=ctx->OpenLookup(name, &status); returns a reference to the object named by the Composite Name "name" relative to the context "ctx". If "name" is equal to "NULLNAME", "OpenLookup" returns an object reference to the object "ctx". (See FIG. 4 and 5.)

Bind

Usage: ctx->OpenBind(name, objref, flags, &status); binds the terminal atomic part of the supplied Name "name" to the supplied object reference. The binding is made in the penultimate context in "name", relative to "ctx". If "name" is Atomic, the binding is done in the context "ctx". Unless the "BIND_SUPERSEDE" flag is set, the binding must be exclusive. That is, the operation returns an error code in "status" (see the section "Status Information" below) and the operation fails if the terminal Atomic part of the provided Name is already bound. If the "BIND_SUPERSEDE" flag is set, then the bind operation will overwrite any existing binding.

Unbind

Usage: ctx->OpenUnbind (name, &status); removes the binding of the terminal Atomic part of "name" from the penultimate context of "name" relative to "ctx".

List Usage: listing=ctx->OpenList(name, &status); returns a list (of type "NameList") of all of the names that are bound in the context named by "name" taken relative to "ctx".

Status Information

In the preferred embodiment of the federation OpenFederation, Status Information is handled in the following manner.

Status Structure

Each of the operations returns a parameter "status", which is a structure (i.e. a set of data). The field "status.err" contains a "status code" of type "statuscode". The interpretation of the remainder of the status structure depends on this status code. Except in those cases specified differently below, the interpretation is as follows: "status.where" contains the name of an intermediate context up to which the operation proceeded normally. (The name in "status.where" is interpreted relative to the "returning" context. That is, if a context "ctx" obtains the status information from a lower-level context service, then "ctx" must revise "status.where" so that it names the same object relative to "ctx".) The field "status.ref" will always be an object reference to the context named by "status.where"; "status.rest" will contain a name, which relative to "status.where" is equivalent to the input name. If the boolean field "status.precisely" is true, then the returning context guarantees that "status.where" names precisely the context in which an error occurred. If "status.precisely" is false, the returning context guarantees only that "status.where" names an intermediate context up to which no error occurred during that operation, but not necessarily the context in which the error occurred.

Status Codes

In the preferred embodiment, the following Status Codes are defined:

"OK"—the operation succeeded. For all operations except "OpenList", "status.where" will contain the penultimate prefix of "name", (so "status.rest" will contain the terminal atomic part of the input parameter "name"). For the "OpenList" operation, "status.where" will be the entire "name", and "status.rest" will be "NULLNAME". For all operations "status.precisely" will always be "true" on an "OK" return. "NO_PERMISSION"—permission was not granted for some required operation either at a terminal or intermediate context.

"ALREADY_BOUND"—(this code applies to the operation "bind" only), the terminal atomic name had an existing binding in the penultimate context, and the "BIND_SUPERSEDE" flag was not set.

"NOT_CONTEXT"—either the OpenList operation was invoked on an object that was not a context, or the resolution of the supplied name reached a non-context object before reaching the penultimate context.

"NOT_FOUND"—the terminal atomic name was not bound in the penultimate context or resolution could not proceed beyond some intermediate context because the next atomic name requiring resolution was not bound.

"ILLEGAL_NAME"—some component of the name was not a well-formed name in the associated naming system.

"UNSUPPORTED_OP"—the operation invoked was not supported by the terminal context. This can happen on "OpenBind", "OpenUnbind", and "OpenList" operations. As indicated above, all federation members, by definition must support the CompositeLookup operation (in the preferred embodiment "OpenLookUp"), but may choose not to support the operations of "OpenBind", "OpenUnbind", and "OpenList". If they do not support these operations, then this Status Code would be returned whenever one of those operations is invoked. In the preferred embodiment, all of these operations are implemented.

Exceptions

In the preferred embodiment, an OpenFederation context may raise an exception if it is unable to return normally with a status described accurately by one of the status codes. Normally these exceptions would be masked by intermediate OpenFederation contexts before reaching a caller outside the OpenFederation federation. It is therefore required that federation members have a minimal recovery procedure that every federation context must follow before allowing an exception to proceed to its immediate caller. In the preferred embodiment, every OpenFederation context must follow such a minimal recovery procedure before allowing an exception to proceed to its immediate caller, upon receipt of, for example, the following status code:.

"_FAILURE"—an unexpected failure prevented normal return.

As can be seen from the above description of the preferred embodiment of a federation of disparate naming systems, the minimal requirements for joining a similar federation include:

1. An agreement on the general definition of a Composite Name syntax for use by the federation. The composite name will be a sequence of one or more names, $n_1, n_2, \ldots, n_k$ from possibly disparate Naming Systems. The preferred Composition syntax is as described above for the OpenFederation embodiment. Other composition rules may be used which are equivalent.

2. A set of operating means for performing the function of Composite Name lookup at a minimum. It is also suggested that the operations of binding, unbinding, and listing names also be implemented.

From the above it can be seen that new members can be added to the federation without impact to the existing members. That is, adding a new Naming System to an existing Federation only requires that a reference to a Context in the new Naming System be bound to a Name in a Context in one or more of the prior member Naming Systems.

Thus it can be seen that in such a system any number of disparate naming systems may be federated with minimal implementation cost to the individual systems and clients, and that no intermediate customized agents

or gateways are required, resulting in better performance and fewer fault-tolerance issues.

As this invention may be embodied in several forms without departing from the spirit of the essential characteristics thereof, the present embodiment is therefore illustrative and not restrictive, since the scope of the invention is defined by the appended claims rather than by the description preceding them, and all changes that fall within the metes and bounds thereof are therefore intended to be embraced by the claims.

I claim:

1. In a distributed computing environment, an apparatus for a federated naming system which can resolve a composite name composed of any number of disparate naming systems, said apparatus comprising:

a) a plurality of arbitrary naming systems, each of said naming systems comprising a plurality of names, cash name being comprised of one or more atomic names, said atomic names being pointers to corresponding objects, each of said naming systems having names with syntax unknown to other naming systems, thereby permitting a new naming system to be added to said distributed computing environment without having to change any other associated naming system;

b) a composite name, comprising a sequence of one or more names from different naming systems which are connected as a sequence of components, each component comprising a naming system identification symbol and a name from said naming system, said naming system identification symbol being uniquely related to one of said plurality of arbitrary naming systems, said sequence of components having an arbitrary number of said components corresponding to said one or more names;

c) a first lookup device for resolving said composite name into said composite name's corresponding parts by returning a head name and a tail name to a request that said composite name bc resolved, and if said taft name is a non-null value said head name is openfive to point to a context where said tail name can be further resolved, and repeating said resolving operation iteratiycly until said tail name returned is a null value at which time a location of an object designated by said composite name has been determined; and

d) a second lookup device which is local to an individual one of said plurality of arbitrary naming systems which is used to resolve names unique to said individual naming system, into associated atomic names and to return a pointer to an object designed by an atomic name therby providing to said first lookup device a resolution for a name in said individual naming system.

2. The apparatus claimed in claim 1 wherein said first lookup device comprises a third logic mechanism operative in a first context to provide an object reference if the composite name is an atomic name; said third logic mechanism being operative to provide a reference to a context object in another naming system a tail of the composite name, if the composite name refers to a context object in said another naming system; and said third logic mechanism being operative to provide a tail of said composite name within a naming system in said first context.

3. The apparatus defined in claim 2 wherein said third logic device is operative to perform operations of composite name binding, unbinding, and listing names.

4. The apparatus defined in claim 3 wherein the apparatus is operative in an object oriented distributed computing system.

5. In a distributed computing environment, a computer implemented method for a federated naming system which can resolve a composite name composed of any number of disparate naming systems, said method comprising the steps of:

a) identifying a plurality of arbitrary naming systems, each of said naming systems comprising a plurality of names, each name being comprised of one or more atomic names, said atomic names being pointers to corresponding objects, each of said naming systems having names with syntax unknown to any other naming system, thereby permitting a new naming system to be added to said distributed computing environment without requiring changes to any other existing naming system in said environment;

b) identifying a composite name, comprising a sequence of components, each component comprising a naming system identification symbol and a name from said naming system, said naming system identification symbol being uniquely related to one of said plurality of arbitrary naming systems, said sequence of components having an arbitrary number of said components wherein said composite name is a sequence of nested addresses;

c) using a first lookup device, resolving said composite name into said composite name's corresponding parts by returning a head name and a tail name to a request that said composite name be resolved, and if said tail name is a non-null value stud head name is operative to point to a context where said tail name can be further resolved, and repeating said resolving operation iteratively until said tail name returned is a null value, at which the a location of an object designated by said composite name has been determined; and

d) using a second lookup device which is local to an individual one of said plurality of arbitrary naming systems which is used to resolve names unique to said individual naming system into associated atomic names and to return a pointer to an object designated by an atomic name providing to said first lookup device a resolution for a name in said individual naming system.

6. The method described in claim 5 wherein the step of resolving the composite name comprises the steps of establishing a composite lookup operation which is operative in a first context to provide an object reference if the composite name is an atomic name, said composite lookup operation being operative to provide a reference to another naming system context and a tail of the composite name, if the composite name refers to another naming system.

7. In a distributed computing system, a computer implemented method for a federated naming system comprising the steps of:

a) providing a plurality of naming systems, each of said naming systems comprising a plurality of names, each name being comprised of one or more atomic names, said atomic names being pointers to corresponding objects, each of said naming systems having names with syntax unknown to any other naming system, thereby permitting a new naming system to be tided to said distributed computing

system without requiring any changes in any other naming system in said computing system;

b) combining names from said naming systems into a composite name by

i) designating a first prefix name represented by a naming system identification symbol to identify a first naming system;

ii) appending said first prefix name to a name in said first naming system, said name and appended first prefix name designated a first naming system component name; and

c) using a left-to-right order for combining said first naming system component name to a sequence of similar naming system component names corresponding to an arbitrary number of said plurality of naming systems to form said composite name; and

d) providing a plurality of operations for operating on composite names including using a first lookup operation to resolve a composite name into a reference to an object harnell by said composite name relative to a specified context, by returning a head name and a tail name to a request that said composite name be resolved, and if said tail name is a non-null value said head name is operative to point to a context where said tail name can be further resolved, and repeating said resolving operation iteratively until said tail name returned is a null value, at which time a location of an object designated by said composite name has been determined, and using a second lookup device which is local to an individual one of said plurality of arbitrary naming systems which is used to resolve names unique to said individual naming system into associated atomic names and to return a pointer to an object designated by an atomic name providing to said first lookup device a resolution for a name in said individual naming system.

8. The method as recited in claim 7 wherein said step of providing a plurality of operations for operating on composite names comprises the additional steps of:

a) providing a bind operation for binding an atomic name portion of a component name to an object reference in a context, said object reference being supplied as a parameter in an invocation of said bind operation and said context being a context of a naming system to which said component name belongs;

b) providing an unbind operation for removing a binding of an atomic name portion of a component name to an object reference from a context; and

c) providing a list operation for providing a list in a specified context, of all names and the corresponding object references.

9. A computer system comprising:

a) a plurality of naming systems, each of said naming systems having a naming syntax which is unknown to any other naming system, thereby permitting a new naming system to be added to said computer system without requiring changes in any existing naming system, each naming system comprising:

i) a plurality of names, each name being comprised of one or more atomic names, said atomic names being pointers to corresponding objects;

ii) a first lookup device for resolving a composite name and operative to return a head name and a tail name in response to any request for a name to be resolved, and if said tail name is a non-null value said head name is operative to point to a context where said tail name can be further resolved, and repeating said resolving operation iteratively until said tail name returned is a null value, at which time a location of an object designated by said composite name has been determined; and

iii) a second lookup device for resolving a name unique to said naming system into said name's corresponding atomic names and to return a pointer to an object designated by at least one said corresponding atomic name; and

b) a composite name created by combining a first name from a first naming system with a second name from a second naming system by attaching a prefix to each of said first and second names, said prefix designating said corresponding first or second naming system, and combining an arbitrary number of said naming system identifier prefix and naming system name combinations in a left-to-right order corresponding to a hierarchy of said naming systems wherein said composite name is a sequence of one or more names which constitute an address of a particular object.

* * * * *

EXHIBIT N

5,903,889 - Filed: June 9, 1997 Issued: May 11, 1999 System and method for translating, collecting and archiving patient records	Google AutoLink
1. A computer system with a plurality of data records on a plurality of databases, and a standardized format for addressing said data records, said computer system comprising:	Google maintains a plurality of data records on multiple databases. There is a standard format for addressing these records, for example book records may contain an ISBN number, which have a standardized format. In Exhibit 3-A of Google's responses and objections to plaintiff's first set of interrogatories (nos. 1-15), hereafter referred to as Google's responses, it is alleged that Google AutoLink does not provide a plurality of data records on multiple databases since a web page is not a data record. However, the World Wide Web itself is a database, whose components are web sites. These websites consist of web pages which are the fields of the data record arranged in a particularly defined structure, i.e., the interlinking of one web page within a site to other web pages within the website. This characterization of a web page as a data record meets the definition of data record established by Judge Crabb in Opinion and Order 02-C-0647-C in the case HyperPhrase Technologies, LLC and HyperPhrase, Inc., v. Microsoft Corporation. This opinion and order was the judge' ruling on the construction of certain claims in the HyperPhrase patents involved with that suit. This definition arose in consideration of claims 49 and 53 of the '461 patent. The '461 patent was combined with the '889 patent in the '298 patent. In Google's responses it is also alleged that there is not a standardized format for addressing the alleged data records in the plurality of databases and that, moreover, Google does not provide or have control over the claimed "computer system". The Google Toolbar Help, which I accessed on 10/21/2006, states "The online review of a great new restaurant has the place's address but no map. You could type the restaurant's street, city, and ZIP code into the search box, but why bother, when clicking the Toolbar's AutoLink button will automatically create a link to an online map (US addresses only)? AutoLink can also link package tracking numbers to delivery status, VIN numbers (US) to vehicle history, and publication ISBN numbers to Amazon.com listings." Another web page provided by Google through Google Toolbar Help, also accessed on 10/21/2006, states "How do I use AutoLink? The AutoLink feature adds link to the page you're viewing if it recognizes certain types of information on the page. For example, AutoLink will link a U.S. street address to a map of that address or the tracking number of a package to a status page for your package. AutoLink also recognizes car VIN numbers and book ISBN numbers. [new paragraph] Click on the "AutoLink" button to create a link on a page, or click on the arrow to the right of the "AutoLink" button and choose a link from the drop down menu."

5,903,889 - Filed: June 9, 1997 Issued: May 11, 1999 System and method for translating, collecting and archiving patient records	Google AutoLink
	Each of these links will also address a record using a standardized format. For example, ISBN numbers are not always expressed with an identical format. In order to achieve the AutoLink functionality that Google claims on this web page, it is necessary both to have, whether explicitly or implicitly a standardized format, e.g., for the ISBN number, and control over a computer processing device, which may be referred to as a computer system, to identify two or more text strings in the data record, i.e., the web page, which differ in format, as being ISBN numbers, which have a standardized format. I reserve that right to modify my opinion on this point, as well as any other points in my expert testimony as more information becomes available to me either through documents provided by Google or through my own subsequent efforts.
(a) a user interface having an interactive display program for requesting one of said data records and displaying a plurality of interface supported data formats;	Google provides a user interface at the Google home page for searching Google's databases, including, for example, New York Times book reviews. A variety of web pages are displayed. In Google's response Google contends that Google's homepage does not display a plurality of interface supported as required by this claim. However, as accessed on 10/21/2006, Google's homepage did display a plurality of interface supported data formats. For example, Images, Video, News, Maps, and Preferences. The user interface of Google's homepage is certainly interactive as it allows users to enter search terms or to click on hyperlinks. In response to both of these user actions the program responds at which point the user may take further action and so on. Google's interface is viewed through any widely available web browser, such as Internet Explorer or Mozilla FireFox. Google's toolbar also operates in such web browsers.
(b) means for receiving a reference to a first data record from said interactive display program;	The Google search engine will send the appropriate URL to the user's machine. In this example, the engine will provide a reference, which includes a hyperlink, which is a URL, to a book review from the New York Times for the book *Amazon.com for Dummies.* Google contends in Google's response that Google AutoLink does not provide a means for receiving a reference to a first data record from said interactive display program. However, Google does provide a means for receiving a reference to a first data record via its interactive display program, both in its homepage and in the user's browser. When a user clicks on the

5,903,889 - Filed: Jun Issued: May 11, 1999 System and method fo, collecting and archiving patient records	Google AutoLink
	URL, the hyperlink, which is part of the reference displayed by the Google home page, which is contained within an interactive browser, e.g., Internet Explorer or Mozilla FireFox, Google uses this functionality to receive the reference, i.e., the book review from the New York Times web site. **Corresponding Structure**: The corresponding structure from the '889 patent is the data transmission and collection system 110 executing algorithm steps described at col. 7, ll. 51-56. These lines state that the data requests may be in the form of a URL with optional data fields sent with it to assist in identifying the record to be received. See also Fig. 12A, box 540. The Google homepage can also use the URL in an identical way to receive the reference. **Identical/Equivalent Analysis**: This use of a URL by the Google search engine is an identical structure according to my present understanding.
(c) means for retriev: data record;	The New York Times server sends the web page in response to the request for the first record prepared by Google (and selected by the user's click). The user's browser, including AutoLink, then retrieves the record; the first record is information about the specific book including the phrase "ISBN" plus the ISBN number. **Corresponding Structure**: The corresponding structure from the '889 patent is the data transmission and collection system 110 executing step 564 described at col. 8, ll. 25-27. These lines state that after the database has produced the requested data record that the record may be received by the data translation and collection system for additional processing. See also Fig. 12A, box 564. Google homepage also uses the URL in an identical way to retrieve the reference. The Google search engine uses this same structure, i.e., a URL, to send the first data record, e.g., a book review from the New York Times web site to the user.

5,903,889 - Filed: June 9, 1997 Issued: May 11, 1999 System and method for translating, collecting and archiving patient records	Google AutoLink
	Identical/Equivalent Analysis: This use of a URL by the Google search engine is an identical structure according to my present understanding.
(d) means for parsing said first data record to identify a reference to a second data record;	Google AutoLink feature parses the retrieved record to find "ISBN" followed by a 10 digit numbers – a reference to additional specific book information which is available, for example, from Amazon. Google contends in Google's response that Google AutoLink does not satisfy this limitation for three reasons. First, Google contends that a web page is not a data record. This contention has already been addressed above. Second, Google contends that Google AutoLink does not identify references to a second data record. However, according to the Google documentation quoted twice already above, AutoLink does identify references to second data references. In particular "AutoLink also recognizes car VIN numbers and book ISBN numbers." These ISBN and VIN numbers are references, which are key words or phrases, as indicated at col. 8, ll. 37-40 of the '889 patent, as discussed in the next paragraph. **Corresponding Structure**: The corresponding structure from the '889 patent is the data transmission and collection system 110 executing step 570 described at col. 8, ll. 37-40. These lines state that the first data record is parsed to locate data references, e.g., hypertext links, multi-media requests, and key words or phrases. According to the same AutoLink reference cited above from Google's web site, "The online review of a great new restaurant has the place's address but no map. You could type the restaurant's street, city, and ZIP code into the search box, but why bother, when clicking the Toolbar's AutoLink button will automatically create a link to an online map (US addresses only)? AutoLink can also link package tracking numbers to delivery status, VIN numbers (US) to vehicle history, and publication ISBN numbers to Amazon.com listings." As described above, to achieve this linking functionality AutoLink would need to parse the first data record, i.e., the book review from the New York Times website. Merely finding a character string equal to "ISBN" or text string which is a 10 digit number would not lead to an accurate linking to a second data record. Rather, as does the system claimed in the '889 patent, it is necessary to identify an initial data reference, here

5,903,889 - Filed: June 9, issued: May 11, 1999 System and method for trallecting and archiving patient records	Google AutoLink
	ISBN, followed by a second modifying reference, here the 10 digit number. **Identical/Equivalent Analysis**: This use of parsing by the Google AutoLink feature is an identical structure according to my present understanding.
(e) means for modifying ce to said second data record to create an address, said ng operable to retrieve said second data record; and	Google AutoLink modifies the reference to a second record (the 10 digit ISBN number) to create an address. For an ISBN number the address http://www.amazon.com/exec/obidos/ASIN/0764558404/104-0193822-0551125, is built, which can be used to retrieve the second record from Amazon. Google in Google's response contends both that Google AutoLink does not perform the function of modifying a record to create an address and also that the pattern of text recognized by Google AutoLink is not a reference to a data record. This contention that Google AutoLink does not perform the function of modifying a record to create an address is refuted both by the earlier quote from Google. Toolbar Help, "How do I use AutoLink? The AutoLink feature adds link to the page you're viewing if it recognizes certain types of information on the page. For example, AutoLink will link a U.S. street address to a map of that address or the tracking number of a package to a status page for your package. AutoLink also recognizes car VIN numbers and book ISBN numbers. [new paragraph] Click on the "AutoLink" button to create a link on a page, or click on the arrow to the right of the "AutoLink" button and choose a link from the drop down menu." and by the example discussed in the paragraph immediately preceding, which shows the modification of a reference to a second data record to create an address which is operable to retrieve the second data record. The contention that the pattern of text recognized by Google AutoLink is not a reference to a data record is refuted by the same argument made in (d) where it was observed that ISBN and Vin numbers are references, which are key words or phrases, as indicated at col. 8, ll. 37-40 of the '889 patent. **Corresponding Structure**: The corresponding structure from the ''889 patent is the data transmission and collection system 110 executing steps 598-600 described at col. 9, ll. 1-11. These lines state that .in

5,903,889 - Filed: June 9, 1997 Issued: May 11, 1999 System and method for translating, collecting and archiving patient records	Google AutoLink
	step 598 the data translation system 110 uses the Hypertext Cipher 138 to convert any text portion of the selected data record into a browser compatible format, such as HML format and that in step 600 the data translation system 110 inserts hypertext links or other references in accordance with the hypertext cipher and also states that the record may be interpreted and modified or reformatted. It is my contention that Google AutoLink modifies the reference to a second data record found in the first record and makes the address operable to retrieve the second data record. According to the '889 patent, col. 7, ll. 12-15, for each data type the hypertext cipher 138 uses special instructions or codes to modify the record. For example one hypertext cipher would be need for ISBN numbers; another for VINs. **Identical/Equivalent Analysis**: It is my opinion that Google AutoLink would need to use this identical structure to achieve the same functionality.
(f) means for sending said modified first data record to said interactive display program.	The first record is displayed on the browser with the ISBN number converted to a data retrieval link. Google contends in Google' response that Google AutoLink does not perform the function of sending a modified first data record and that AutoLink does not modify web pages. The contention that AutoLink does not modify web pages is already addressed in (e). The already quoted text from Google Toolbar Help also clearly states "Click on the "AutoLink" button to create a link on a page . . ." Thus AutoLink creates a hyperlink in the web page and sends it to the user through the interactive display program, the browser. **Corresponding Structure**: The corresponding structure from the '889 patent is the data transmission and collection system 110 executing step 604 described at col. 9, ll. 12-15. These lines state that the data collection and translation system 110 forwards the record to the requesting workstation or processor. As described in (b) both the system 110 of the '889 patent and the Google search engine will use a URL to send the modified record to the user's browser.

5,903,889 - Filed: June 9, 1996: May 11, 1999 System and method for transacting and archiving patient records	Google AutoLink
	Identical/Equivalent Analysis: This use of a URL by the Google search engine is an identical structure according to my present understanding.
7. The computer system of claim said reference to said second data record comprises phrase.	The reference to the second data record comprises the keyword phrase "ISBN" plus the ISBN number, which includes the keyword "ISBN". Google contends in its response that Google AutoLink does not satisfy this limitation because it does not reference second data records and because the references identified by the Plaintiffs do not comprise a keyword phrase. This first contention has already been answered above in (d). However, following the construction of claims ruling of Judge Crabb in HyperPhrase v. Microsoft at 19 a "keyword phrase" means a recognized text string that serves as the hypertext link. Thus, following the examples quoted above taken from Google Toolbar Help, Google AutoLink does use keyword phrases, e.g., "ISBN" or "VIN".

6,516,321 - Filed: August 13, 1999 Issued: February 4, 2003 Method for database address specification	Google AutoLink and Google Toolbar
1. A method for identifying a referenced record referenced in a referencing record wherein the referenced record is referenced in the referencing record by at least a combination including a data reference (DR) and a modifier reference (MR), the method comprising the steps of:	Google AutoLink uses web pages as referencing records. Google AutoLink identifies ISBN numbers in web pages. These ISBN numbers in turn correspond to referenced records. The referenced record is referenced by a data reference (the text "ISBN") and a modifier reference (a 10 digit ISBN number, for example "0764558404 ") Google's response contends that AutoLink does not identify a referenced record in a referencing record, that web pages are not records and that web pages do not contain references to records. Google's response further contends that Google AutoLink does not use a combination of DR and MR. These contentions have already been answered above. In particular the ISBN example just given is the use of a combination of a DR and an MR.
(i) receiving the referencing record;	The web page is received by the personal computer running Google AutoLink. Google's response contends that a web page is not a referencing record. As discussed above, a web page is a record and it contains DRs which reference other records, as discussed above.
(ii) analyzing the referencing record to identify a DR, when a DR is identified:	The Google AutoLink computer searches the web page for the text "ISBN". This identifies a DR ("ISBN"). Google's response contends that it does not identify DRs to records and that it does none of the steps (a) through (c), but that even if Plaintiff's views of the claim is accepted, that it does not do the steps in the same order. As discussed above in answer to Google's earlier contentions, DRS to records are identified by AutoLink. Furthermore, the order in which these steps are performed have a logical sequencing that must be followed.
(a) identifying an MR rule set (MRRS) specifying the relationship between an MR and the DR;	The Google AutoLink computer uses the rule that ISBN numbers are 10 digits long and are placed following an ISBN DR. Google's response contends that AutoLink does not use an MRRS to specify a relationship between an MR and a DR, nor does it use a DR to identify a group of possible records or an MR to identify a particular record in a group. However, the example given shows that this does indeed take place, either literally or in a substantially similar way.
(b) analyzing the referencing record in accordance with the	The Google AutoLink computer analyzes the record to see if a 10 digit number follows the

6,516,321 - Filed: August 13, 1999 Issued: February 4, 2003 Method for database address specification	Google AutoLink and Google Toolbar
MRRS to identify the existence of the MR and, when the MR is identified;	DR text "ISBN". The result is an identification of the existence of an MR. Google's response contends that Google AutoLink does not use an MRRS to identify the existence of an MR, but the example just discussed shows that this does indeed take place, either literally or in a substantially similar way.
(c) identifying the referenced record associated with the DR/MR combination.	The Google AutoLink computer identifies the record referenced by "ISBN: 0764558404", which will be the address http://www.amazon.com/exec/obidos/ASIN/0764558404/104-0193822-0551125. Google's response contends that Google AutoLink does not use an MRRS to identify the referenced record associated with the DR/MR combination, but the example just discussed shows that this does indeed take place, either literally or in a substantially similar way.
27. A method to be used with a rule set including subject matter specific tag pairs and corresponding search rules, a separate tag pair for each of a plurality of different information types and a separate search rule for each pair, each pair including a begin tag and an end tag, the method comprising the steps of:	Google AutoLink can recognize ISBN, FedEx, VIN numbers, and addresses using separate search rules for each and will place a different tag pair around the found text. Different search rules are applied for each information type, and different tag pairs are also applied to each information type. Google's response contends that Google AutoLink: a) does not provide a method to be used with a rule set, including subject matter specific tag pairs and corresponding search rules; b) does not provide a separate tag pair for each of a plurality of data types nor a separate search rule for each data type. Google's response further contends that Google AutoLink does none of the steps (a) through (c), but that even if Plaintiff's views of the claim is accepted, that it does not do the steps in the same order. So far, as discussed above, I have been given no access to Google documents that would address the contention that Google AutoLink does not use subject matter specific tag pairs, then it is my view that Google AutoLink uses a substantially similar way to achieve the same functionality. For example, it might use subject matter specific offsets to achieve the same purpose. Even if these steps are different because they involve the use of a substantially similar technique, there would still be performed in the same logical sequencing. In addition, the tag pairs actually used by Google appear to be subject matter specific.
(a) receiving a record;	The Google AutoLink computer receives a copy of the web page from the server using the browser software.
(b) examining the record according to the search rules to identify	The Google AutoLink computer examines the web page, using a variety of search rules to

6,516,321 - Filed: August 13, 1999 Issued: February 4, 2003 Method for database address specification	Google AutoLink and Google Toolbar
record segments including information of each of the information types;	identify ISBN, FedEx, VIN number or addresses. One example is to examine the text of the web page to identify a record segment which is an ISBN number.
(c) when a record segment is identified which is of a particular information type:	For example, when an ISBN number is found proceeded by "ISBN", the ISBN number is identified as an ISBN number (i.e. as a record segment of the type ISBN number).
accessing the tag pair associated with the information type; inserting the begin tag before the identified segment and inserting the end tag after the identified segment.	The identified segment is the ISBN Number. An appropriate tag pair is accessed is placed around the ISBN number, with the begin tag before the identified segment (the number) and the end tag after the number.
24. The method of claim 1 further including the step of linking the record reference to the referenced record.	The ISBN Number is hyperlinked to Amazon, or the appropriate link is provided to the appropriate destination. AutoLink and the Toolbar perform the linking step. Google's response contends that Google AutoLink does not infringe this claim for the reasons given for non-infringement of claim 1 and also because it does not perform the step of linking the record reference to the referenced record. These contentions have been answered in discussion of claim 1 and earlier several times in discussion of the example taken from Google Toolbar Help.
86. A method for use with an application wherein specifying references (SRs) in one record to other records which are selectable to access the other records are visually distinguished from other record information so as to indicate selectability, the method also for use with a system which enables a user to designate and also select SRs where designation comprises pointing to an SR without selection and, wherein a seemingly general SR is modified by other record information which renders the SR relatively specific, the method for indicating the specific nature of an SR prior to selection and comprising the steps of:	The Google Toolbar is an application program that works with the Google AutoLink computer where specifying references in a web page (the referencing record) are identified and linked to second records. For example the text "40967812221" corresponds to another record stored in an Amazon.com database. The ISBN number is visually distinguished by different coloring and underling (like a hyperlink) and is shown to be selectable. The application program enables the user to designate an ISBN number (by hovering over it) and select it (by clicking on the number). The ISBN number is a general 10 digit number that is modified by other record information (in this case the proceeding text "ISBN"). This modified information allows the toolbar to determine that the number isn't merely a ten digit telephone number or other ten digit number. Google's response contends that Google AutoLink does not satisfy the "record" requirement of this claim, nor the "specifying references", nor that the so-called specifying references are selectable. Google's response further contends that Google AutoLink does not provide a method for indicating the specific nature of an SR prior to selection. So far, as discussed above, I have reviewed no Google documents that would address this contention about Google AutoLink. However, in light of the example discussed above, it is my view that Google AutoLink uses a substantially similar way to achieve the same functionality, if it

6,516,321 - Filed: Au?9 Issued: February 4, 2003 Method for database cification	Google AutoLink and Google Toolbar
	does not literally satisfy these requirements.
when an SR is designting the specific nature of the SR.	When an ISBN number is designated (by hovering over it) a message is displayed by the Google toolbar application program that this is a Google Toolbar AutoLink corresponding to the ISBN number of a book.